НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»

УДК

«До захисту допущено»

Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
“ ” 20***20*** р.

Магістерська дисертація

зі спеціальності: *123. Комп'ютерна інженерія*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *123. Комп'ютерні системи та мережі*

на тему: Спосіб розпізнавання фейкових новин у мережі Інтернет

Виконав (-ла): студент (-ка) 6 курсу, групи ІО-81мн
(шифр групи)

Міщенко Людмила Дмитрівна _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник проф. дтн. Сімоненко В.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент доц. каф. СПС, к.т.н., Орлова М. М. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*

(повна назва)

Рівень вищої освіти – другий (магістерський) за науковою програмою

Спеціальність *123. Комп'ютерна інженерія*
(код і назва)

Спеціалізація *123. Комп'ютерні системи та мережі*_____

(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
« » 20***20*** р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Міщенко Людмила Дмитрівна
(прізвище, ім'я, по батькові)

1. Тема дисертації Спосіб розпізнавання фейкових новин у мережі інтернет

Науковий керівник дисертації проф. дтн Сімоненко Валерій Павлович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___» _____ *2020* р. №

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження: алгоритм пошуку та аналізу фейкових новин

4. Предмет дослідження: способи та засоби створення алгоритму пошуку та аналізатора фейкових новин.

5. Перелік завдань, які потрібно розробити: за необхідною методологією розробки програмного забезпечення виконати огляд існуючих рішень; на основі результатів аналізу існуючих програмних рішень спроектувати

алгоритм пошуку та аналізу фейкових новин; виконати тестування розробленого програмного продукту.

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	д.т.н., проф. Кулаков Ю. О.		

7. Дата видачі завдання: _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Огляд існуючих програмних рішень		
2.	Аналіз існуючих програмних рішень		
3.	Проектування програмного продукту		
4.	Розробка програмного продукту		
5.	Тестування програмного продукту		
6.	Розробка стартап-проекту		
7.	Оформлення документації		

Студент _____ Міщенко Л. Д.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Сімоненко В. П.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Спосіб розпізнавання фейкових новин у мережі Інтернет

студентом: Міщенко Людмилою Дмитрівною

Робота складається із вступу та чотирьох розділів. Загальний обсяг роботи: 79 аркушів основного тексту, 15 ілюстрацій, 10 таблиць. При підготовці використовувалася література з 23 різних джерел.

Актуальність. Дедалі більшу популярність набуває використання різних новинних веб ресурсів у мережі Інтернет та соціальних мережах для поширення інформації. Набираючи деяку аудиторію читачів та користуючись їх довірою, такі джерела починають розповсюджувати фейкові новини чи маніпуляції. Тому ідея захисту населення від дезінформації та поширення маніпулятивного впливу в умовах війни є вкрай гострою й необхідною в сьогоденні.

Використання сучасних технологій є необхідним фактором у боротьбі із поширенням фейкових даних. При чому, основна задача полягає в швидкому автоматичному аналізі інформації, а також розповсюдженні спростувань та правдивих фактів. Тому розробка нових алгоритмів пошуку та аналізу потоку новин являється вкрай актуальною задачею.

Мета і завдання дослідження. Метою магістерської роботи є підвищення ефективності системи для виявлення, синтезу та аналізу новин шляхом застосування технології Natural Language Processing.

Для досягнення мети дослідження поставлено і вирішено такі завдання:

- дослідження структури найпопулярніших новинних веб ресурсів із фейковими новинами;
- збір та структуризація правдивої інформації та спростованих фактів;
- розробка програмного забезпечення для збору та синтезу інформації;
- розробка програмного забезпечення для аналізу текстів;
- ілюстрація роботи програми та аналіз отриманих результатів.

Об’єкт дослідження – процес пошуку та аналізу фейкових новин.

Предмет дослідження – способи створення алгоритмів синтезу та аналізу

фейкових новин.

Методи досліджень. Для досягнення поставлених в магістерській роботі задач, використано методи теорії графів, методи технології Natural Language Processing.

Наукова новизна одержаних результатів роботи полягає у наступному:

- запропоновано спосіб аналізу та перевірки на правдивість інформації в мережі Інтернет та соціальних мережах;
- поєднано технологію Natural Language Processing, відстань Левенштейна та коефіцієнт семантичної схожості слів TF-IDF у розробленому програмному продукті для збору, синтезу та аналізу новин на фейковість.

Проведене дослідження дає можливість використання розробленого програмного забезпечення для класифікації новини на правду, маніпуляцію чи фейк та застосування його в соціальній мережі Facebook чи як розширення у браузері Google Chrome.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до вирішення задачі аналізу текстів за допомогою технології Natural Language Processing. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Практична цінність. Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- вдосконалення методів аналізу текстів заданого формату;
- фактчекінг із автоматизованими інформаційними технологіями.

Публікації:

Міщенко Л. Д. СПОСІБ РОЗПІЗНАВАННЯ ФЕЙКОВИХ НОВИН. *Science, society, education: topical issues and development prospects*: V International Scientific and Practical Conference Kharkiv, Ukraine. 12-14 April 2020

Liudmyla Mishchenko, Valerii Simonenko. METHOD FOR DETECTING FAKE NEWS BASED ON NATURAL LANGUAGE PROCESSING. Міжнародна науково-технічна конференція “The International Conference on Security, Fault Tolerance, Intelligence” (ICSFTI2020 Online).

Ключові слова

Технологія Natural Language Processing, фейк, маніпуляція, аналіз тексту, відстань Левенштейна, коефіцієнт семантичної схожості слів та словосполучень TF-IDF.

ABSTRACT

Method for Detecting Fake News on the Internet

student: Liudmyla Mishchenko

The work consists of an introduction and four sections. Total workload: 79 pages of main text, 15 illustrations, 10 tables. In preparation, literature from 23 different sources was used.

Topicality. The use of various news web resources on the Internet and social networks to spread information is becoming increasingly popular. By recruiting and trusting some readers, such sources begin to spread fake news or manipulation. Therefore, the idea of protecting the population from misinformation and spreading manipulative influence in times of war is extremely urgent and necessary today.

The use of modern technologies is a necessary factor in the fight against the spread of fake data. Moreover, the main task is the rapid automatic analysis of information, as well as the dissemination of denials and true facts. Therefore, developing new algorithms for searching and analyzing the news flow is an urgent task.

The purpose and objectives of the study. The aim of the master's thesis is to increase the efficiency of the system for the detection, synthesis and analysis of news through the use of technology Natural Language Processing.

To achieve the goal of the study set and solved the following tasks:

- research on the structure of the most popular news web resources with fake news;
- collecting and structuring truthful information and simplified facts;
- development of software for the collection and synthesis of information;
- development of text analysis software;
- illustration of the program and analysis of the results.

The object of study is the process of detecting and analyzing fake news.

The subject of the study - methods to create algorithm for fake news synthesis

and analysis.

Research Methods. To achieve the goals set in the master's work, the methods of graph theory, natural language processing techniques were used.

The scientific novelty of the results obtained is as follows:

- a method of analyzing and verifying the truth of information on the Internet and social networks is proposed;
- combines Natural Language Processing technology, Levenstein distance, and the TF-IDF semantic similarity factor in the developed software to capture, synthesize, and analyze fake news.

The research made it possible to use the developed software to classify news into truth, manipulation or fake and to apply it on the social network Facebook or as an extension in the Google Chrome browser.

Personal contribution of the applicant. The master's research is a self-completed work that reflects the author's personal approach and personally obtained theoretical and applied results related to solving the problem of text analysis using Natural Language Processing technology. The purpose and objectives of the study were formulated jointly with the scientific supervisor.

Practical value. The results obtained can be used in future research in the following areas:

- improving the methods of text analysis of a given format;
- fact-checking with automated information technology.

Posts:

Mishchenko LD Method for Detecting Fake News on the Internet. Science, Society, Education: Topical Issues and Development Prospects: In International Scientific and Practical Conference Kharkiv, Ukraine. 12-14 April 2020

Liudmyla Mishchenko, Valerii Simonenko. METHOD FOR DETECTING FAKE NEWS BASED ON NATURAL LANGUAGE PROCESSING. Міжнародна науково-технічна конференція “The International Conference on Security, Fault Tolerance, Intelligence” (ICSFTI2020 Online).

Keywords

Natural Language Processing technology, fake, manipulation, text analysis, Levenstein distance, TF-IDF semantic similarity of words and phrases.

ЗМІСТ

ВСТУП

У сучасному світі телебачення та інформаційні канали все більше відходять на другий план, у той час коли поширення новин та інформації відбувається за допомогою мережі Інтернет і соціальних мереж. Користувачі стають підсвідомими жертвами поширення дезінформації та маніпуляції в мережі Інтернет.

В умовах інформаційної війни надзвичайно важливо критично сприймати й аналізувати інформацію. Але при високому темпі життя та постійного використання гаджетів користувачі не бажають витрачати час на декількаразову перевірку фактів. Тому застосування мобільного додатку чи розширення для веб браузера є дуже актуальними на сьогодні та може значно полегшити процес перевірки фактів, а також прояснити правдиві дедалі новин для користувачів, які не зацікавлені в додатковій перевірці прочитаної інформації чи витрати часу на фактчекінг.

Для розробки такого програмного продукту необхідно використовувати сучасні технології, які допоможуть швидко аналізувати та спростовувати факти. Одним із найкращих варіантів аналізу тексту є автоматичний синтез природної мови.

Обробка природної мови використовуються у всіх галузях сучасних технологій. У більшості технологічних рішень розпізнавання і обробка «людських» мов давно впроваджена: саме тому звичайний IVR з жорстко заданими опціями відповідей поступово відходить у минуле, чатботи починають все більш адекватно спілкуватися без участі живого оператора, фільтри в пошті працюють на основі автоматизованих процесів без людської допомоги і т.д. Тому й процес перевірки фактів, який завжди потребує людські ресурси (журналістика), можливо пришвидшити та вдосконалити за допомогою Natural Language Processing.

Natural Language Processing (далі - NLP) - обробка природної мови - підрозділ інформатики і AI, присвячений комп’ютерному аналізу природної

(людські) мови. NLP дозволяє застосовувати алгоритми машинного навчання для тексту й мови.

NLP можливо використовувати, щоб створювати системи на кшталт розпізнавання мови, узагальнення документів, машинного перекладу, виявлення спаму, розпізнавання іменованих сутностей, відповідей на питання, автокомплітора, інтелектуального введення тексту, перевірка на орфографічні та граматичні помилки в тексті та інше.

Сьогодні майже 95% смартфонів мають вбудовану функцію розпізнавання мови - в них використовується NLP для того, щоб розуміти людську мову. Також більшість гаджетів, таких як ноутбуки, планшети, використовуються з вбудованим в ОС розпізнаванням природної мови (Cortana - Windows, Siri - MacOS, ...).

Метою роботи є розпізнавання записаної мови, тобто тексту, із використанням сучасних технологій. Автоматична обробка та аналіз зібраних даних на фейкові або правдиві факти чи маніпуляцію. А також прискорення поширення спростувань та зменшення розповсюдження дезінформації в мережі Інтернет.

Використання сучасних технологій є необхідним фактором у боротьбі із поширенням фейкових даних. При чому, основна задача полягає в швидкому автоматичному аналізі інформації, а також розповсюдженні спростувань та правдивих фактів. Тому розробка нових алгоритмів пошуку та аналізу потоку новин являється вкрай актуальною задачею.

Стек технологій вибрано на основі мови програмування Python із бібліотеками BeautifulSoup для збору даних із різних текстових джерел та веб ресурсів, NLTK для роботи із Natural Language Processing та аналізу тексту. Зібраний контент записується та зберігається в нереляційній базі даних MongoDB. Швидкий пошук у базі даних реалізовується за допомогою ElasticSearch, який також відповідає за збереження токенів та релевантність видачі результатів від токена з найважомішим семантичним навантаженням до найменш вагомого. На основі мови програмування Python також реалізовано

алгоритм обрахунку відстані Левенштейна, який необхідний для порівняння текстів. Також застосовується вбудований модуль NumPy для вирахування коефіцієнта схожості змістовного значення тексів, який приймає значення скорингу за TF-IDF.

РОЗДІЛ 1
АНАЛІЗ ПРОГРАМНИХ ПРОДУКТІВ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ NATURAL LANGUAGE PROCESSING ТА МЕТОДІВ РОЗПІЗНАВАННЯ ФЕЙКОВИХ НОВИН

1.1. Відомі рішення автоматичного спростування фейкової інформації

1.1.1. Розширення для веб браузера Fake news guard

Цей програмний продукт є основним конкурентом з точки зору комерційного продажу розробки. Проте він базується виключно на порівнянні фактів із бази даних, наповненої журналістськими дослідженнями. Або перевіряє джерело інформації на надійність.

У Fake news guard розробляються інструменти, щоб допомогти виявити неправдиву інформацію набагато більшими темпами, ніж дослідник людини може зробити самостійно [1].

Рис. 1.1. Логотип програмного продукту Fake news guard для спростування неправдивих новин

Проте розробники цього продукту ґрунтуються виключно на самостійній перевірці чи інформація є достовірною, особливі програмні технології не залучаються. Така система спирається на інформацію зібрану людиною, яка перевіряє факти. Іноді журналіст використовує штучний інтелект для проведення смислових порівнянь.

У Fake news guard доступне розширення для веб браузера Chrome, яке допомагає користуватися системою під час навігації по веб сторінках. Та можете пасивно контролювати відвідані сторінки або фейсбук-стрічку користувача. Користувач може активно надсилати для перевірки підозрілі посилання чи джерела.

Це Google Ghrome розширення забезпечує пасивну форму захисту. Кожна сторінка, яку відвідує користувач, і кожне посилання, яке згадане на його стрічці Facebook, перевіряється так званим “чорним списком”. Цей список містить електронні ресурси, які постійно випускають дезінформації як зброю інформаційної війни або намагаються привабити аудиторію гучними заголовками, без підтвердження фактів. Якщо джерело статті було виявлено в чорному списку, то користувач отримує повідомлення про споживання неправдивої інформації у спеціально розробленому веб-переглядачі червоного кольору.

1.1.2. Ресурс Snopes для перевірки фактів в Інтернеті

Рис. 1.2. Логотип незалежного видання Snopes спростувань неправдивої інформації

Snopes.com - незалежне видання, яке належить Snopes Media Group. Snopes розпочав свою діяльність у 1994 році, досліджуючи міські легенди, містифікацію та фольклор. Засновник Девід Міккелсон, згодом приєднався до публікацій в Інтернеті, задовго до того, коли більшість людей були підключені до Інтернету. По мірі зростання попиту на достовірні перевірки фактів, зростав і Snopes. Зараз це найстаріший і найбільший веб-сайт, що перевіряє факти, широко розцінений журналістами, фольклористами та читачами як неоціненний супутник дослідження [2].

Проте перевірка фактів Snopes.com ґрунтується на доказах журналістів та виключно контекстуалізованому методі спростування. Тобто тільки слідча група залучена до перевірка фактів, ніяких програмних технологій. Також журналісти документують свої посилання на джерела, щоб читачі мали право робити незалежні дослідження та складати власну думку.

Тобто, даний програмний продукт не є конкурентом для запропонованого способу перевірки та спростування фактів у мережі Інтернет. Адже швидкість автоматичної обробки інформації значно швидша, тобто відповідає сучасним темпам поширення масової інформації.

1.2. Найпоширеніші комерційні програмні продукти для спростування нішових фейків

У наш час купівля онлайн набуває все більших обертів. На віртуальному ринку продаються та купуються всі необхідні для життя товари. Але вибір необхідних речей для користувача ґрунтується виключно на характеристиці товару та опублікованих відгуках інших про нього.

Для клієнтів, продавців чи рецензентентів відгуки про продукцію мають надзвичайно важливе значення. Для продавця вони допомагають покращити рейтинг, а для замовника - відгуки допомагають приймати рішення про купівлю. Але іноді деякі відгуки про обраний варіант є підробками.

Підроблений огляд - це позитивний (зазвичай 5-зірковий) відгук, залишений для товару тим, хто навіть не купив цей товар, але діє або від імені

продавця, або продавець використовує інше ім’я користувача. Підроблені відгуки - це негативне явище на онлайн ринку, оскільки вони малюють помилкову картину товару і спонукають клієнтів купувати те, що може бути субстандартним продуктом. По суті, використовуючи помилкову рекламу, зловмисники користуються умовами онлайн торгівлі та довірою покупців.

Звичайно користувач може і самі помітити підроблені огляди чи відгуки. Проте є кілька зручних інструментів, які допоможуть зробити перевірку автоматично. Таку технологію застосовують всесвітньо відомі інтернет магазини Amazon, Ebay, Sephora та інші.

1.2.1. Аналізатор відгуків Fakespot

Fakespot - це інструмент аналітики, створений спеціально для аналізу та обробки оглядів продуктів на Amazon, Ebay та інших веб-сайтах, що займаються покупками. Користувач просто копіює посилання товару в пошукову систему сайту, а вбудована, захищена технологія потім аналізує огляди з метою виявлення повторних слів і фраз, частоти, дат, схем придбання та перевірених покупок, щоб скласти список відгуків, які ймовірно є підробки. Fakespot доступний безкоштовно для використання через вбудований зручний онлайн-інструмент на їхньому сайті.

1.2.2. Reviewmeta - інструмент для аналізу та фільтрування відгуків у онлайн магазинах

Reviewmeta аналізує мільйони оглядів, щоб допомогти користувачам вибрати правдиві. Цей інструмент простий у використанні. Достатньо скопіювати та вставити посилання на товар у інструмент пошуку сайту, і система створить результати "Передати", "Попередити" або не допоможе керувати рішенням покупця. Тобто не матиме необхідних спростувань у базі. Система обґрунтовує свої висновки незалежно від мови написання чи кількості оглядів, а також швидкості отримання списку відгуків та кількості відгуків, які не

підтверджені. Для зручності Reviewmeta.com доступний для ноутбуків та ipads, а також для ряду програм для телефонів.

1.2.3. Інструмент для огляду резюме та відгуків Thereviewindex

Ще один зручний інструмент в Інтернеті, The**review**index аналізує достовірності відгуків про продукт з використанням посилання на нього. Спочатку збираються всі раніше згадані дані про продукт. Потім система створює підсумок загальної кількості оглядів та окремо кількість не підтверджених. До результатів також включає письмове резюме із зазначенням того, чи не повторювані виявлені зразки, які можуть підказувати помилкові огляди. Хоча трохи базовіше, ніж Fakespot.com та Reviewmeta.com, це все-таки зручний ресурс для виявлення фальшивих відгуків.

Рис. 1.3. Логотипи інструментів для перевірки відгуків у онлайн магазинах

Ці інструменти прості та звучні у використанні, проте вони мають дуже обмежений функціонал та направлені виключно на перевірку відгуків та рецензій у онлайн магазинах. І хоча змістовно вони не несуть конкурентного значення, проте сам алгоритм перевірки все ж ближчий до запропонованого рішення, ніж звичайна перевірка людськими ресурсами.

1.3. Порівняльний аналіз існуючих методів перевірки фактів

У таблиці 1.1. порівняно існуючі методи автоматичної перевірки інформації на правдивість. На основі цього аналізу можна зробити висновок, що

запропонований метод має значні переваги завдяки обраним методам та стеку технологій для його реалізації.

Таблиця 1.1.

Порівняльний аналіз існуючих способів автоматичного спростування фейкової інформації

Назва продукту	Сфера застосування	Ціль застосування	Методи реалізації ідеї
Fake news guard	Стрічка новин у Facebook або у веб-браузері Google Chrome	Спростування фейкових новин	Часткова автоматизація – порівняння джерела новини із списком сайтів-поширювачів файків, журналістське розслідування
Snopes	Веб-сайт для журналістів, фольклористів та інших читачів	Поширення достовірних і перевірених фактів	Контекстуалізований метод спростування фейків, тобто перевірка інформації виключно журналістами
Fakespot	Онлайн магазини (Amazon, Ebay, …)	Аналіз та обробка відгуків та оглядів продуктів у інтернет магазинах	Технологія аналізу підроблених відгуків шляхом виявлення повторних слів, фраз, схем придбання покупок та інше.
Reviewmeta	Онлайн покупки зі смартфонів,	Аналіз оглядів товарів та	Порівняння вхідного огляду з уже

	планшетів, рідше ноутбуків.	видалення фейкових	існуючою вибіркою фейкових
The**review**index	Інтернет магазини	Аналіз достовірності відгуків про товар	Пошук однакових відгуків на різні товари
FakesRadar (Factology System)	Стрічка новин у Facebook чи розширення веб-браузера Google Chrome	Виявлення фейкових новин, боротьба з дезінформацією	Автоматичний аналіз вхідної статті за допомогою NLP

1.4. Огляд існуючих комерційних програмних продуктів для аналізу природного мовлення.

1.4.1. Cortana

Рис. 1.3. Логотип віртуального помічника Cortana

Віртуальний помічник Cortana, який розпізнає мову, розроблений компанією Microsoft, використовується в операційній системі Windows. За допомогою Cortana ОС розпізнає та аналізує людську мову та виконує запити в браузері, керує додатками смартфону. Тобто користувач може створювати нагадування, відкривати додатки, відправляти листи, грати в ігри, дізнаватися погоду і т.д.

1.4.2. Siri.

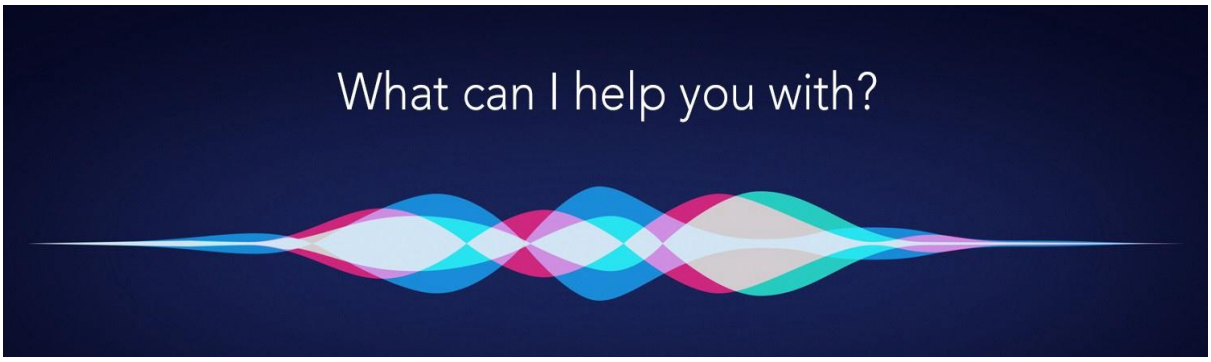

Рис. 1.4. Логотип віртуального помічника Siri

Siri - це помічник для ОС від Apple: iOS, watchOS, macOS, HomePod і tvOS. Безліч функцій також працює через голосове управління: зателефонувати / написати кому-небудь, відправити лист, встановити таймер, зробити фото і т.д.

Цей додаток спілкується природною мовою, щоб відповідати на питання і давати рекомендації. Тобто, реалізовано не тільки розпізнавання та обробка людської мови, а й застосовано технологію Natural Language Processing для підтримки зворотної комунікації з користувачем.

1.4.3. Gmail

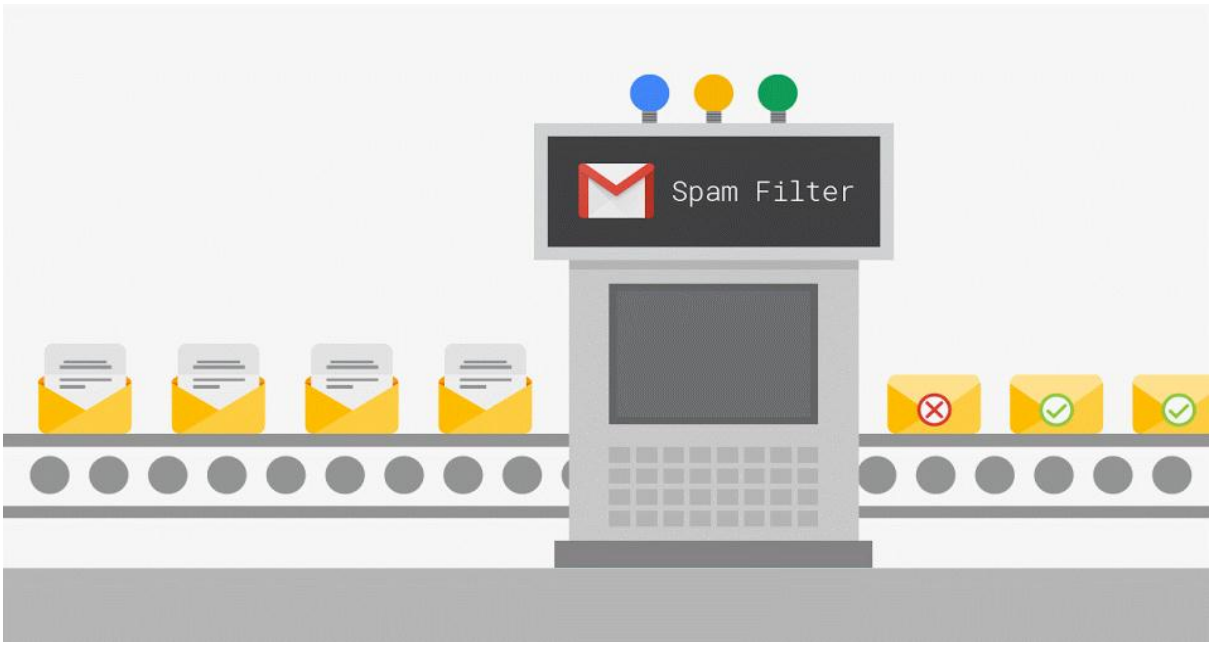

Рис. 1.5. Схематичне зображення роботи віртуального помічника Gmail

Відомий поштовий сервіс вміє визначати спам, щоб він не потрапляв у вхідні поштової скриньки користувача. У основі принципу цієї технології закладено парсинг та аналіз контенту листів.

1.4.4. Dialogflow

Рис. 1.6. Логотип віртуального помічника Dialogflow

Платформа від Google, яка дозволяє створювати NLP-ботів. Наприклад, можна зробити бота для замовлення піци, якому не потрібен старомодний IVR, щоб прийняти та зробити синтез замовлення користувача.

Висновки до розділу 1

Перший розділ присвячений аналізу програмних продуктів із використанням технології Natural Language Processing та вже відомі методи розпізнавання фейкових новин. Розглянуто приклади успішного впровадження лідерами ринку рішень з використанням NLP, що дозволить скласти картину сучасного стану технології та можливості вирішення поставлених завдань на її основі.

Зокрема:

1. Проаналізовано відомі рішення автоматичного спростування фейкової інформації. А саме технічні інструменти, які використовуються в їх реалізації для перевірки фактів, також їх аудиторії, маркетингові моделі та способи розвитку продукту.

2. Проведено дослідження стосовно комерційного розвитку програмних продуктів із вбудованим автоматичним аналізом новин. Показано, що вони не є конкурентоспроможними, адже такі продукти як Fakespot, Reviewmeta та The**review**index є вузьконаправленими та розвинуті лише в напрямку перевірки відгуків та описів товарів у інтернет магазинах. А в таких продуктах як Fake news guard та Snopes не реалізований процес автоматичної перевірки фейкової інформації. Тобто вся перевірка грунтується виключно на ручній перевірці фактів фактчекерами.

3. Проведено дослідження сучасного стану розвитку технології NLP у мобільних застосунках, розглянуто основні структурні елементи Natural language processing та проблеми, які дозволяють повністю або частково вирішення впровадження цієї технології.

4. Визначено основні переваги та недоліки програмного забезпечення для розпізнавання та аналізу природного мовлення чи текстів.

5. Аргументовано необхідність впровадження програмного забезпечення для швидкого розпізнавання та аналізу новин чи фактів. Впровадження цього підходу сприятиме позитивному впливу на процес

поширення правдивої інформації в умовах інформаційної війни та, зокрема, поширення пандемії.

6. Розглянуто способи застосування технології NLP у комерційних програмних продуктах, вказані їхні переваги, недоліки та виділено ті способи, що можуть ефективно бути використані в побудові антифейкового програмного забезпечення.

РОЗДІЛ 2
ІНСТРУМЕНТИ ТА ІСНУЮЧІ РІШЕННЯ NLP В МОВІ ПРОГРАМУВАННЯ PYTHON ДЛЯ РОЗРОБКИ АНАЛІЗАТОРА ПРИРОДНЬОЇ МОВИ

2.1. Інструменти в мові програмування Python для розробки аналізатора природньої мови.

Створення програмного забезпечення для роботи із природною людською мовою доволі клопітний процес. Реалізація можлива різними мовами програмування із використання різноманітних інструментів, фреймворків, бібліотек, модулів і т.і. Найбільш розвинені такі інструменти в мові програмування Python, тому вона вважається найкращою для рішення запропонованої проблематики.

Для мови програмування Python розроблено бібліотеку NLTK - Natural Language Toolkit (з англійської - набір інструментів з природних мов), яка містить реалізовані та вбудовані інструменти, необхідні для розробки аналізатора природного мовлення. Актуальна на даний момент версія - NLTK 3.

NLTK - це провідна платформа для побудови програм на Python для роботи з людськими мовами. Він надає прості у використанні інтерфейси для понад 50 корпоративних та лексичних ресурсів, таких як WordNet, а також набір бібліотек для обробки тексту для класифікації, токенізації, стримування, розмітки тегів, розбору та семантичних міркувань, обгортки для промислових NLP бібліотек, та активний дискусійний форум.

Бібліотека NLTK зручна та проста у використанні. Завдяки вичерпній документації API, а також практичному посібнику, який зрозумілий для розробників із базовими основами програмування та граматики в лінгвістиці, NLTK підходить як для лінгвістів, інженерів, студентів, викладачів, дослідників, так і для користувачів інших галузей. NLTK доступний для Windows, Mac OS X та Linux. Найкраще, що NLTK - це безкоштовний проект з відкритим кодом, підтримкою якого постійно займаються розробники бібліотеки.

NLTK називають «чудовим інструментом для навчання та роботи в комп'ютерній лінгвістиці за допомогою Python» та «дивовижною бібліотекою для гри з природною мовою» [3].

Natural Language Processing за допомогою Python має велике практичне значення в програмуванні для обробки природньої мови. Реалізований інструмент NLTK орієнтований на написання програм мовою Python, роботу з модулями, категоризацію тексту, аналіз мовної структури тощо. Він чудово підходить для виділення основних слів чи словосполучень у тексті, за допомогою яких можна реалізувати пошук подібної текстової інформації.

2.1.1. Токенізація за реченнями.

Токенізація (іноді - сегментація) за реченнями - це процес поділу писемної мови на речення-компоненти. Ідея виглядає досить просто. В англійській і деяких інших мовах можливо виокремлювати речення кожен раз, коли знаходимо певний знак пунктуації, наприклад, крапка, кома, дефіс та інше.

Але, навіть в англійській мові ця задача нетривіальна, так як, наприклад, крапка використовується і в скороченнях. Таблиця скорочень може значно допомогти під час обробки тексту, щоб уникнути невірної розстановки меж у реченнях. У більшості випадків для цього використовуються додаткові бібліотеки чи модулі, які можуть допомогти вирішити поставлену задачу завдяки більш детальній реалізації технології.

Для прикладу роботи цього методу взято невеликий текст англійською мовою:

"*Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East. It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.*"

Щоб зробити токенізацію запропонованого вище тексту за допомогою, наприклад, бібліотеки NLTK, необхідно скористатися вбудованим методом nltk.sent_tokenize.

```
text = "Backgammon is one of the oldest known board games. Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East. It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice."
sentences = nltk.sent_tokenize(text)

for sentence in sentences:
    print(sentence)
    print()
```

Результатом виконання програми буде 3 окремих речення:

“Backgammon is one of the oldest known board games.

Its history can be traced back nearly 5,000 years to archeological discoveries in the Middle East.

It is a two player game where each player has fifteen checkers which move between twenty-four points according to the roll of two dice.”

2.1.2. Токенізація за словами

Токенізація (іноді - сегментація) за словами - це процес поділу речення на слова-компоненти. В англійській і багатьох інших мовах, що використовують ту чи іншу версію латинського алфавіту, пробіл - це явний роздільник слів.

Проте, можуть виникнути проблеми, якщо використовуватиметься тільки пробіл. До слова, в англійській мові складові іменники пишуться по-різному, іноді через пробіл. У цьому випадку необхідно підключати додаткові бібліотеки чи інструменти.

Для прикладу використаємо вище згадане речення *text* та застосуємо метод `nltk.word_tokenize` із зазначеної бібліотеки NLTK [4].

```
sentences = nltk.word_tokenize(text)
for sentence in sentences:
    print(sentence)
```

```
print()
```

Результат виконання програми наступний:

['Backgammon', 'is', 'one', 'of', 'the', 'oldest', 'known', 'board', 'games', '.']

['Its', 'history', 'can', 'be', 'traced', 'back', 'nearly', '5,000', 'years', 'to', 'archeological', 'discoveries', 'in', 'the', 'Middle', 'East', '.']

['It', 'is', 'a', 'two', 'player', 'game', 'where', 'each', 'player', 'has', 'fifteen', 'checkers', 'which', 'move', 'between', 'twenty-four', 'points', 'according', 'to', 'the', 'roll', 'of', 'two', 'dice', '.']

2.1.3. Лематизація та стемінг тексту

Зазвичай тексти містять різні граматичні форми одного і того ж слова, а також можуть зустрічатися однокореневі слова. Лематизації і стемінг мають на меті привести всі словоформи до однієї, нормальної словникової форми.

Для цього методу наведено приклад із перетворенням різних форм слова до однієї:

dog, dogs, dog's, dogs' => dog

Те ж саме, але застосоване до цілого речення:

the boy's dogs are different sizes => the boy dog be differ size

Лематизації і стемінг - це окремі випадки нормалізації і вони відрізняються.

Стемінг - це грубий евристичний процес, який відрізає «зайве» від кореня слів, часто це призводить до втрати словотворчих суфіксів чи префіксів.

Лематизації - це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб у результаті перетворити слово до його канонічної формі - лемме.

Відмінність в тому, що Стеммер (конкретна реалізація алгоритму стемінг) працює без "знання" контексту і, відповідно, не розрізняє різницю між словами, які мають різний зміст в залежності від частини мови [5].

Однак у Стеммер є і свої переваги:

- значно простіший у використанні та впровадженні;
- реалізована програма працює відчутно швидше;
- більш низька «акуратність», тобто виділення ресурсів на змістовне значення слова, може не мати жодного значення в деяких випадках.

Приклади:

1. Слово good - це лема для слова better. Стеммер не побачить цей зв'язок, тому що тут потрібно перевіряти словником.
2. Слово play - це базова форма слова playing. Тут впораються і стемінг, і лематизації.
3. Слово meeting може бути як звичайною формою іменника, так і формою дієслова to meet, в залежності від контексту. На відміну від стемінг, лематизація спробує вибрати правильну лемму, спираючись на контекст.

Приклад програмного коду, який реалізує цей метод, мовою Python з бібліотекою NLTK [5]:

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet


def compare_stemmer_and_lemmatizer(stemmer, lemmatizer, word, pos):
    """
    Print the results of stemmind and lemmitization using the passed stemmer, lemmatizer, word and pos (part of speech)
    """

    print("Stemmer:", stemmer.stem(word))
    print("Lemmatizer:", lemmatizer.lemmatize(word, pos))
    print()

lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
compare_stemmer_and_lemmatizer(stemmer, lemmatizer, word = "seen", pos = wordnet.VERB)
```

```
compare_stemmer_and_lemmatizer(stemmer, lemmatizer, word = "drove", pos = wordnet.VERB)
```

Результати виконання програми:

Stemmer: seen

Lemmatizer: see

Stemmer: drove

Lemmatizer: drive

2.1.4. Стоп-слова

Стоп-слова - це слова, які викидаються з тексту до / після його обробки. При застосуванні машинного навчання до текстів, такі слова можуть додати багато шуму, тому необхідно позбавлятися від нерелевантних слів.

Стоп-словами зазвичай виступають артиклі, вигуки, сполучники і т.д., які не несуть змістового навантаження. При цьому треба розуміти, що не існує універсального списку стоп-слів, все залежить від конкретного випадку.

У NLTK є попередньо встановлений список стоп-слів. Перед першим використанням необхідно його завантажити: nltk.download ("stopwords"). Після скачування можна імпортувати пакет stopwords і подивитися на самі слова [6]:

```
from nltk.corpus import stopwords
print(stopwords.words("english"))
```

Результати роботи програми [6]:

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
```

```
'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own',
'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don',
"don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y',
'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn',
"doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn',
"isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't",
'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren',
"weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Нижче описано програмну реалізацію способа, який видаляє стоп-слово із речення. Таким чином очищає вибірку слів від шуму та зайвого навантаження.

```
stop_words = set(stopwords.words("english"))
sentence = "Backgammon is one of the oldest known board games."

words = nltk.word_tokenize(sentence)
without_stop_words = [word for word in words if not word in stop_words]
print(without_stop_words)
```

Результат:

['Backgammon', 'one', 'oldest', 'known', 'board', 'games', '.']

2.1.5. Corpus Reader Objects

NLTK включає різноманітний набір колекцій, які можна застосувати, використовуючи пакет nltk.corpus. Для кожної колекції можна отримати вбудований об'єкт "зчитування з колекції" з nltk.corpus методами.

Більшість колекцій складаються з набору файлів, кожен з яких містить власну документацію (або інші фрагменти тексту). Список ідентифікаторів для цих файлів доступний за допомогою методу fileids () зчитувача колекції.

Кожен зчитувач колекції забезпечує різноманітні методи зчитування даних з її вмісту, залежно від формату зазначеної колекції. Наприклад, розроблено окремі методи для роботи зі простим текстом для зчитування корпусу (тобто, тіла введеного тексту) як “сирого” тексту. Таким вважається, колекції списку слів, списку абзаців, списку речень чи словосполучень.

Кожному з цих методів зчитування може бути надано назву елемента одного документа або список назв елементів документа. Коли буде надано список назв елементів документа, то методи для зчитування об'єднують вміст окремих документів.

Якщо методи зчитування викликаються без жодних аргументів, вони зазвичай завантажують усі вхідні дані із документів у єдину колекцію.

Якщо корпус містить файл README, до нього можна отримати доступ за допомогою спеціального методу readme() [7].

2.1.5. Parsed corpora

Однією з найважливіших колекцій у NLTK є Parsed Corpora. Завдяки їй можна реалізувати значну кількість актуальних сьогоденню задач.

З назви колекції зрозуміло, що вона відповідає за інструмент парсингу текстів. Колекція вміщує в собі модуль Treebank, який реалізовує синтаксичний розбір для кожного речення. Пакет даних NLTK включає вибірку Penn Treebank (у treebank), а також Sinica Treebank (у sinica_treebank).

Повна реалізація Penn Treebank доступна до інсталяції безкоштовно, але як окремий модуль. Разом із NLTK його можна налаштувати на завантаження та синтаксичний розбір текстових файлів чи звичайної змінної типу string. Для цього потрібно завантажити пакет даних PTB і в каталог nltk_data / corpora / ptb, розмістити каталоги BROWN та WSJ інсталятора Treebank (також функціонують посилання). Потім використовувати модуль ptb замість Treebank.

Такий інструмент чудово підходить для аналізу потоку новин. Адже текст кожної новини розглядається як окремий файл, з якого формується синтаксичне дерево розбору (рис. 2.1.). При чому, кожному файлу обов’язково присвоюється унікальний id.

Цей інструмент визначає власні категорії текстів. Потім, за визначеною категорією, текст можна знайти у файлі allcats.txt. Пізніше це можна використовувати для фільтрації за жанрами. Які, наприклад, можуть складатися з новин (для статей одного журналу, видання, джерела) та назв підкатегорій

(художня література, гумор, романтика тощо). Важливо зазначити, що всі вхідні дані повинні бути з одного джерела або стандартизовані за однаковими критеріями оформлення матеріалів [7].

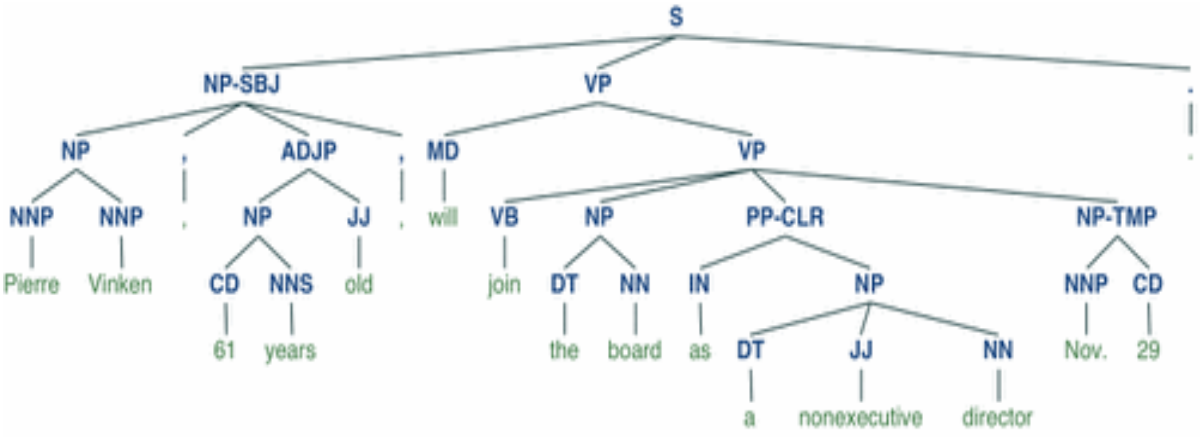

Рис. 2.1. Приклад синтаксичного дерева розбору за допомогою методу Treebank з пакету nltk.corpus

2.1.6. Узагальнене поняття Text Corpus Structure - структура корпусів текстів

Усі тексти можна поділити на так звані корпуси (як названо в офіційній документації до пояснення роботи технології Natural Language Processing), тобто вони відрізняються своєю структурою.

Розрізняють різноманітні структури текстів. Найпростіший вид не має будь-якої структури, тобто це просто набір текстів. Зачасту тексти можна погрупувати у категорії, які, у свою чергу, відповідають певному жанру, джерелу, автору, мові тощо. Іноді ці категорії перетинаються, особливо у випадку з актуальними категоріями, оскільки текст може мати відношення до декількох тем. Інколи колекції тексту мають часову структуру, найпоширенішим прикладом є онлайн чи офлайн видання новин. Саме така структура використовується для реалізації поставленого завдання.

Загальні текстові структури можна поділити, наприклад, на [8]:

- найпростіший вид тексту - це сукупність незалежних текстів без особливої організації;
- жанри - за виділення цієї категорії відповідає метод nltk.brown_corpus;

- тематичні категорії - реалізуються за допомогою методу nltk.reuters_corpus. При чому, нерідко ці категорії можуть перетинатися й утворюються своєрідні об’єднання чи перетин множин;
- вживання текстів за часом - для цієї категорії задіяний метод nltk.inaugural_address_corpus.

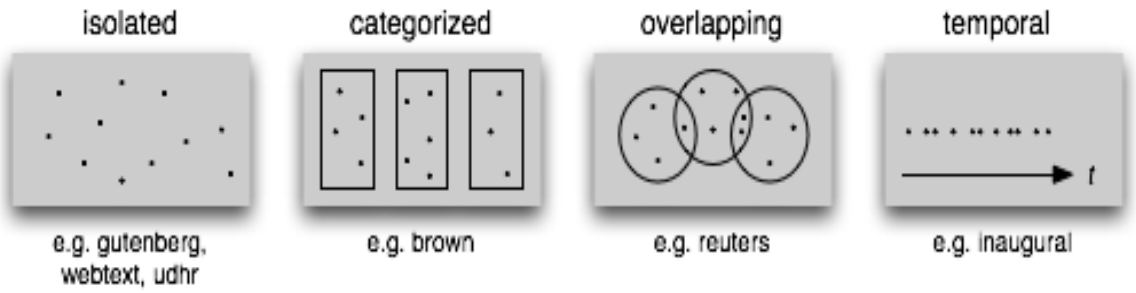

Рис. 2.2. Загальна структура поділу текстів на корпуси

На рис. 2.2. показано загальну структуру поділу текстів на корпуси.

Як видно з малюнка, у першому випадку це незалежні один від одного тексти, для них використовуються такі методи як Gutenbrg, Webtext, Udhr та інші. У другому випадку поділ текстів на категорії, наприклад, за жанрами. Застосовуються методи Brown Corpus і т.і. У третьому квадраті зображено перетини текстів. Для знаходження перетину, наприклад, за темою, використовується метод Reuters Corpus. На останньому квадраті показано часовий парсинг текстів. Найчастіше саме він застосовується до стрічки новин із методом Inaugural Address Corpus [8].

NLTK методи для зчитування текстів та поділу їх на структури, реалізують ефективний доступ до різноманітних так званих корпусів NLTK. Вони використовуються для роботи з новими структурами текстів та надають ширші функціональні можливості для модуля зчитування текстів.

2.2. Парсинг та аналіз тексту із застосуванням регулярних виразів

Рис. 2.3. Загальне позначення регулярних виразів

Регулярні вирази є сильним інструментом для пошуку рядків, перевірки їх на відповідність певному шаблону та іншої подібної роботи. Англомовна назва цього інструмента – *Regular Expressions* чи просто *RegExp*. Інакше кажучи, регулярні вирази – це послідовність символів, яка визначає шаблон пошуку.

Реалізація цього інструмента розрізняється в мовах програмування, але без значних відмінностей.

З точки зору синтаксису, будь-який рядок сам по собі є регулярним виразом. Звідси випливає, що вираз Новина, містить тільки одну відповідність рядку ” Новина”. Регулярні вирази є регістрозалежними, тому рядок “новина” (з маленької літери) вже не відповідатиме виразу вище.

Регулярні вираз зазвичай складаються із спеціальних символів, які треба екранувати, як і в усіх мовах програмування. Список спецсимволів виглядає наступним чином: . ^ $ * + ? { } [ ] | ( ). Екранування здійснюється у звичайний спосіб – додаванням символа екранування “\” перед спецсимволом.

Найчастіше екранування потребують символи на початку чи вкінці регулярного виразу. Усередині набору більша частина спецсимволів не потребує екранування, проте використання \ перед ними не вважатиметься помилкою. Символи “\” і “^”, і, бажано “]” необхідно теж екранувати. Так, [][] означає будь-який із символів “]” чи “[“, тоді як [[]x] – винятково послідовність “[x]”. Незвичайна, на перший погляд, поведінка регулярок із символом “]” насправді визначається відомими правилами, але набагато легше просто екранувати цей

символ, ніж їх запам’ятовувати. Крім цього, екранувати треба символ “-“, він використовується для завдання діапазонів.

Для деяких наборів, що часто використовуються, є спеціальні шаблони. Так, для опису будь-якого пробільного символу (пропуск, табуляція, перенесення рядка) використовується /s, для цифр – /d, для символів латиниці, цифр і підкреслення ”_” – /w [9].

Також за допомогою регулярних виразів є можливість перевірити положення рядка відносно іншого тексту. Так вираз /b означає границю слова, /B – не границю слова, символ ^ – початок тексту, а $ – кінець фразу чи тексту. Усі метасимволи, які використовуються в регулярних виразах, описані нижче в таблицях від 2.1. до 2.4. [10].

Таблиця 2.1.

Символи підстановки

Символ	Значення	Приклад
.	Позначає збіг із будь-яким одиничним символом (буквою, цифрою або знаком)	Виразу "Н." відповідають значення "Н1", "На" Виразу "Н.в" відповідають значення "Н1в", "Нов"
?	Позначає відсутність збігу або одиничний збіг із попереднім символом	Виразу "Но?" відповідають значення "Н", "Но"
+	Позначає принаймні одиничний збіг із попереднім символом	Виразу "Но+" відповідають значення "Но", "Ноо"
*	Позначає відсутність або наявність	Виразу "Н*" відповідають

	збігу з попереднім символом	значення "Н", "Но"
|	Створює умову відповідності "АБО" Використання в кінці виразу недоцільне	Виразу "Н|Но" відповідають значення "Н", "Но"

Таблиця 2.2.

Прив’язки

Символ	Значення	Приклад
^	Позначає збіг із прилеглими символами на початку рядка	Виразу "^10" відповідають значення "10", "100", "10х" Виразу "^10" не відповідають значення "110", "110х"
$	Позначає збіг із прилеглими символами наприкінці рядка	Виразу "10$" відповідають значення "110", "1010" Виразу "10$" не відповідають значення "100", "10х"

Таблиця 2.3.

Групи

Символ	Значення	Приклад
( )	Позначає збіг з узятими в дужки символами в точному порядку в	Виразу "(10)" відповідають значення

	будь-якому місці рядка Також використовується для групування інших виразів	"10", "101", "1011" Виразу "([0-9]|[a-z])" відповідає будь-яка цифра чи маленька літера
[ ]	Позначає збіг з узятими у квадратні дужки символами в будь-якому порядку в будь-якому місці рядка	Виразу "[10]" відповідають значення "012", "123", "202", "120", "210"
-	Створює діапазон символів у квадратних дужках для пошуку збігу в будь-якому місці рядка	Виразу "[0-9]" відповідає будь-яке число від 0 до 9

Таблиця 2.4.

Символ виходу

Символ	Значення	Приклад застосування
\	Указує, що прилеглий символ має інтерпретуватися буквально, а не як метасимвол регулярного виразу	Вираз "\." указує, що прилегла крапка має інтерпретуватися як закінчення речення або десятковий роздільник, а не як символ підстановки. Виразу "216\.239\.32\.34" відповідає значення "216.239.32.34"

Таблиця 2.5.

Найуживаніші примітивні шаблони регулярних виразів

Шаблон	Значення
\ w	одне слово
\ d	одна цифра
\ s	одна прогалина
\ W	одне не слово
\ D	одна не цифра
\ S	одна не прогалина
[Abc]	знаходить будь-який символ із зазначених
[^ Abc]	знаходить будь-який символ, крім зазначених
[A-g]	знаходить символ в проміжку від a до g

Регулярні вирази використовуються для додаткового фільтрування заданого тексту. Наприклад, можна прибрати всі символи, які не є словами. У багатьох випадках пунктуація не потрібна і її легко прибрати за допомогою регулярних виразів.

Модуль re в Python реалізує операції з регулярними виразами. Використовується функція re.sub, щоб замінити всі відповідні до шаблону пошуку значення, на зазначений рядок. Так можна замінити всі “неслова” на прогалини [11].

Приклад реалізації методу мовою Python:

```
import re

sentence = "The development of snowboarding was inspired by skateboarding, sledding, surfing and skiing."
```

```
pattern = r"[^\w]"
print(re.sub(pattern, " ", sentence))
```

Результати роботи програми:

'The development of snowboarding was inspired by skateboarding sledding surfing and skiing '

2.6. TF-IDF

У всіх вище описаних варіантах вирішення задачі синтаксичного розбору текстів є проблема: слова з найбільшою частотою мають, відповідно, найбільшу оцінку. У цих словах може бути не так багато інформаційного виграшу для моделі, як в менш частих словах. Один із способів виправити ситуацію - знижувати оцінку слова, яке часто зустрічається у всіх подібних документах. Це називається технологією TF-IDF.

TF-IDF (скорочення від term frequency - inverse document frequency, з *англ.* визначення частота - обернена частота документа) - це статистична міра для оцінки важливості слова в документі, який є частиною колекції або корпусу NLTK.

Скоринг по TF-IDF зростає пропорційно частоті появи слова в документі, хоча до уваги вхаровується також кількість документів, в яких згадується це слово [12].

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Формула 2.1. Скоринг для слова X в документі Y,

де *tf(x, y)* - частота *x* у *y*, *df(x)* - кількість документів, які містять змінну *x*, *N* - загальна кількість документів.

TF (term frequency - частота слова) - відношення числа згадування слова до загальної кількості слів у документі, як показано у формулі нижче:

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

IDF (inverse document frequency - зворотна частота документа) - інверсія частоти, з якою деяке слово зустрічається в документах колекції. Формула має наступний вигляд:

$$IDF(\boldsymbol{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \boldsymbol{term}\ in\ it}\right)$$

У підсумку, обчислити TF-IDF для слова term можна за загальною формулою:

$$TFIDF(\boldsymbol{term}) = TF(\boldsymbol{term}) * \mathrm{IDF}(\mathbf{term})$$

Для прикладу застосування технології TF-IDF можна показати наступне.

Використовується клас TfidfVectorizer з бібліотеки Sklearn, щоб обчислити TF-IDF [12]. Виконується ця технологія з текстовим повідомленням:

I like this movie, it's funny.

I hate this movie.

This was awesome! I like it.

Nice one. I love it.

Результат виконання програми показано в таблиці 2.6.

Таблиця 2.6.

Значення TF-IDF для обраного тексту

	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0.000000	0.571848	0.000000	0.365003	0.450852	0.000000	0.450852	0.000000	0.000000	0.365003	0.000000
1	0.000000	0.000000	0.702035	0.000000	0.000000	0.000000	0.553492	0.000000	0.000000	0.448100	0.000000
2	0.539445	0.000000	0.000000	0.344321	0.425305	0.000000	0.000000	0.000000	0.000000	0.344321	0.539445
3	0.000000	0.000000	0.000000	0.345783	0.000000	0.541736	0.000000	0.541736	0.541736	0.000000	0.000000

Як видно з результатів, у кожному реченні визначено слова, які несуть найбільше змістовне навантаження. Тобто, обчислення показника TF-IDF є важливою складовою для пошуку фейкових новин на одну й ту ж саму тему, але в різних онлайн джерелах.

Висновки до розділу 2

У рамках другого розділу проведено аналіз найпоширеніших рішень Natural Language Processing, застосованих до текстів у мові програмування Python. Розглянуто основні способи реалізації технології NLP у мові програмування Python, а також інструменти, необхідні для розробки поставленої задачі, а саме:

1. Обрано відкриту бібліотеку NLTK, яка задовольняє поставлені вимоги для аналізу природної мови.

2. Проведено дослідження інструментів токенізації текста за реченнями та словами. Показано доречність застосування деяких так званих “корпусів” бібліотеки NLTK, а саме Brown Corpus, Reuters Corpus, Inaugural Address Corpus, Parsed Corpora та інші.

3. Проаналізовано та доведено ефективність токенізації, лематизаї та стемінгу текстів. Що застосовується для розбиття їх на слова чи словосполучення, а також пізніше їх приведення до нормальної словникової форми.

4. Визначено основні переваги та недоліки використання Regular Expression: за допомогою регулярних виразів можна простіше реалізувати парсинг тексту, який виконується значно скоріше. Проте результати парсингу не матимуть значення частин мови у словах, їх синтаксичне та семантичне значення.

5. Аргументовано необхідність обрахунку значення показника схожості TF-IDF, який є статистичною мірою для оцінки важливості слова в документі.

Спираючись на проведений аналіз та дослідження технології NLP, способів аналізу текстів, необхідно зосередити увагу на розв’язання наступних завдань:

- розробити програму для збору правдивих та перевірених фактів і новин;

- розробити програму для аналізу невідомих новин на правдивість;
- розробити API для роботи фактчекінгового програмного продукту;
- оцінити ефективність запропонованого підходу, виділити переваги та недоліки, а також вказати на сфери можливого застосування запропонованого рішення.

РОЗДІЛ 3
ЗАГАЛЬНА СХЕМА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ФЕЙКОВИХ НОВИН

3.1. Загальний опис алгоритму

Із наведеного вище матеріалу випливає, що застосування технології Natural Language Processing для реалізації запропонованого способу розпізнавання фейкових новин є вкрай доцільною в сьогоденні. Адже ця технологія не використовувалася раніше для цілі боротьби із дезінформацією.

Основним поставленим завданням є розробка способу автоматичного розпізнавання фейкових новин. Тобто запропонований підхід рішення задачі повинен забезпечити розрізнення правдивої інформації від сфабрикованої.

Загальну структуру реалізації програми можна описати із таких підзадач:

1) наповнення бази даних - скрапінг сайтів журналіств-фактчекерів;
2) текстовий аналізатор;
3) перевірка результатів роботи автоматичного аналізу;
4) пошук свіжих фейкових новин на основі вже готової бази даних та сформованих токенів.

Нижче описано короткий виклад алгоритму дій для реалізації запропонованого способу:

1) збір перевірених і підтверджених фактів;
2) упорядкування зібраних даних у базі даних;
3) формування змістовних токенів із текстів та заголовків правдивих статтей;
4) поділ тексту даної статті на слова та формування набору ключових токенів за допомогою технології NLP;
5) порівняння отриманих токенів із заголовка вхідного тексту з токенами заголовків перевірених новин за допомогою алгоритма Левенштейна;
6) порівняння токенів текстів статтей, у яких невелика різниця відстані Левенштейна в заголовку;

7) перевірка результатів.

3.2. Деталізований опис розробки алгоритму та його архітектури

3.2.1. Наповнення бази даних перевіреними фактами

Для збору інформації й формування датасету використовуються тільки перевірені факти та джерела, доступ до яких надають партнери, донори та друзі-фактчеркери.

Одними із найбільших ресурсів для наповнення правдивих новин є бази даних Європейського Союзу та Сполучених Штатів Америки. Ці бази даних містять значний перелік посилань на перевірені джерела та факти, а також підтверджені або спростовані новини.

Такі джерела представлені у вигляді сайтів або надають доступ до власного API. Для збору інформації з веб-сайтів виконується скрапінг за допомогою відкритої Python бібліотеки BeautifulSoup.

Найбільшим сайтом - постачальником даних - являються європейські партнери EUvsDisinfo, які мають більше, ніж 8000 спростованих фейків та перевірених фактів. EUvsDisinfo - це провідний та значущий проект спеціальної групи Європейської служби зовнішніх дій «Схід StratCom». Він був створений у 2015 році для кращого прогнозування, вирішення та реагування на поточні дезінформаційні кампанії та кібератаки Російської Федерації, що стосуються Європейського Союзу, усіх його держав-членів та країн спільного сусідства.

Основна мета EUvsDisinfo - підвищити поінформованість громадськості та розуміння операцій з дезінформації Кремля, а також допомогти громадянам Європи та за її межами розвинути опір цифровій інформації та маніпуляціям ЗМІ [13].

Ще одним вагомим джерелом перевіреної інформації є веб-сайт StopFake. Громадська організація «Центр Медіареформи» – освітня платформа, зас-нована Могилянською школою журналістики з метою запровадження в Украї-ні високих стандартів журналістської освіти, підвищення рівня медіаграмот-ності,

інформування про небезпеки пропаганди та поширення неправдивої інформації (фейків) у ЗМІ. Від початку свого існування ЦМР працює над впро-вадженням західних стандартів журналістики в Україні й підвищенням рівня медіаграмотності різних аудиторій – від студентів і журналістів до пересічних громадян, які прагнуть критично і відповідально споживати медіаконтент. Задля цього команда проводить семінари з фактчекінгу для журналістів, викладачів жур-налістики українських університетів, навчає студентів, блогерів та громад-ських активістів із різних регіонів України та інших країн. Вони тримають руку на пульсі сучасних тенденцій у медіа і долучаються до міжнародних медіа- досліджень. Практикуючі експерти – постійні учасники міжнародних конференцій та форумів, спікери на головних дискусіях про актуальні питання сучасних медіа.

У 2014 році організація «Центр Медіареформи» стала лідером бо-ротьби із пропагандою. А найуспішніший флагманський проект StopFake нині відомий медіапрофесіоналам в усьо-му світі. Він не тільки виявляє випадки поширення неправдивої інформації про події в Україні, а й фак-тично ініціював міжнародну дискусію про те, як про-тистояти такому ганебному явищу [14].

Для наповнення бази використовується іще понад 20 джерел. Наприклад, українська Nepravda.org, грузинському DataCommon.org та інші. Проте всі вони мають різну структуру сайтів та впорядкування даних. Тому скрапінг кожного джерела треба реалізовувати окремим модулем та підлаштовуватися під не схожі стилі викладу інформації різними журналістами.

3.2.2. Упорядкування зібраних даних

Зібрані дані з різних джерел можуть відрізнятися за наповненням, тобто не містити деякі важливі поля бази або дублювати одну й ту ж інформацію. Для цього необхідно провести додаткову оптимізації зі збереженими даними.

Для оптимізації датасету проводиться очистка зібраних даних від дублікатів. Адже при парсингу різних джерел трапляються випадки перевірки

одних і тих же новин декілька разів, що збільшує вимоги до об’ємів пам’яті та значно сповільнює роботу системи.

На першому рівні оптимізації відбувається уніфікація об’єктів за заголовком фейкової новини. Коли видалено всі повторювані заголовки, порівнюється короткий опис до спростування фейка.

Усі записи в зібраному датасеті сортуються за часом, при чому найновіші завжди перші.

3.2.3. Формування змістовних токенів із текстів та заголовків правдивих статтей

Для аналізу тексту використовується технологія Natural Language Processing, яка формує повну змістовну картину тексту. Тобто, відбувається процес формування токенів із отриманих заголовків та текстів.

Вище вказана технологія розбиває текст на токени за універсальними частинами мови [15]. Усі вони наведені в таблиці 3.1.

Таблиця 3.1.

Універсальні частини мови

Тег	**Значення**
ADJ	прикметник
ADP	прийменник
ADV	прислівник
CONJ	сполучник
DET	визначник, артикль
NOUN	іменник
NUM	числівник

PRT	частка
PRON	займенник
VERB	дієслово
.	розділові знаки
X	інше

Заголовок та текст статті аналізуються окремо. Для заголовку отримується список семантичних токенів на основі усіх слів. Адже для заголовка важливо враховувати всі деталі й не втратити жодного слова. Особливо це важливо для боротьби із дезінформацією, де примітивна частка “не” може докорінно змінити зміст тексту фейкової новини чи спростування.

Для тексту окремо формуються списки семантичних та ключових токенів. При цьому під час аналізу з тексту видаляються службові слова, такі як артикль, визначник, частка, розділові знаки тощо.

Процес формування семантичних токенів тексту показано на рисунку 3.1.

На рисунку видно, що технологія NLP будує семантичне дерево розбору. При чому, воно завжди буде бінарним. Результатом розбору повного тексту статті буде набір токенів, які сформувалися на основі семантичного бінарного лісу.

Наступним кроком буде фільтрація отриманих токенів - видалення слів, які не мають ніякого семантичного значення. Таким чином можна зменшити навантаження на систему, обсяги бази даних та значно прискорити результати автоматичного фактчекінгу. Схематичне виконання цього кроку NLP показано на рис. 3.2.

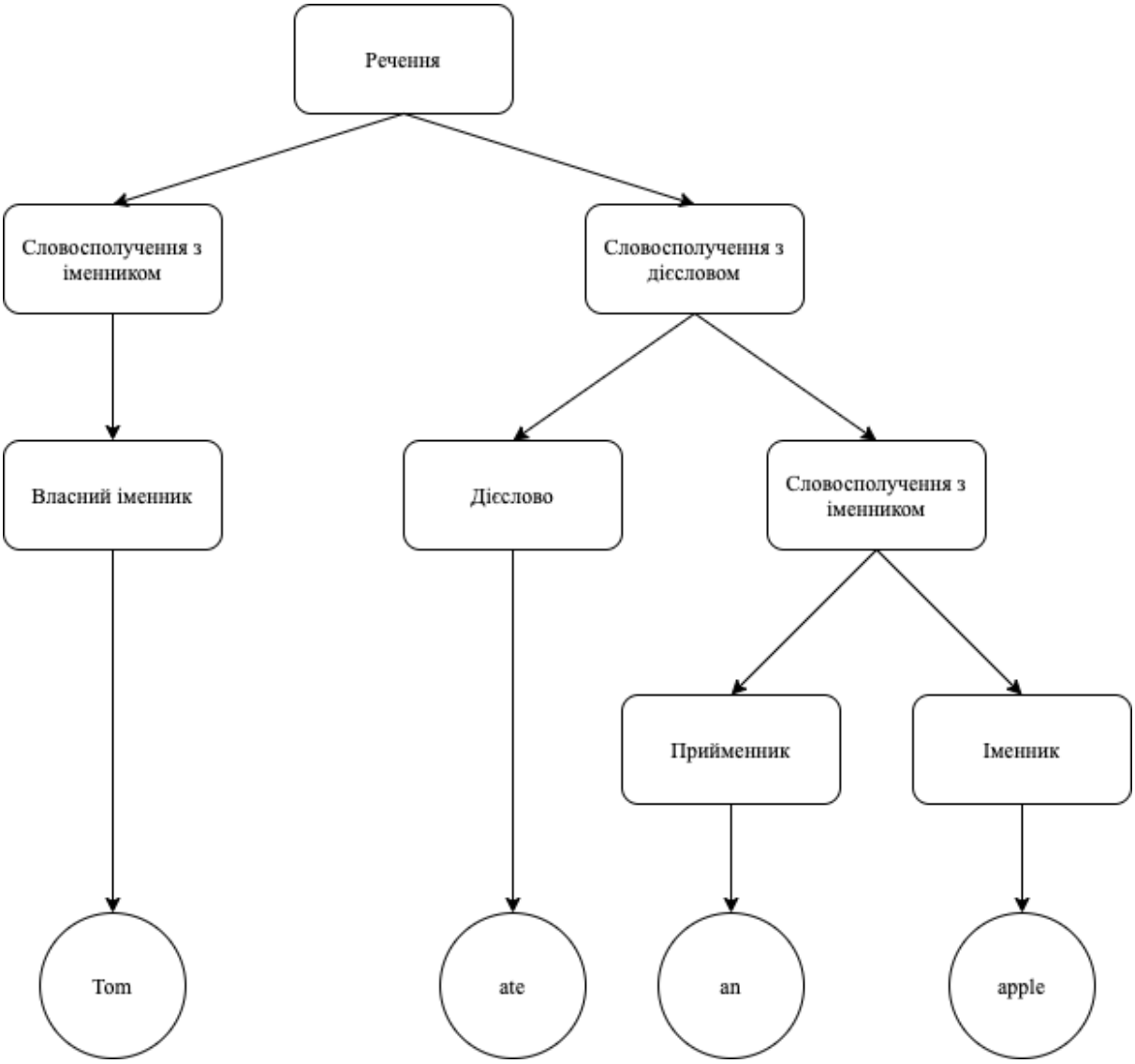

Рис. 3.1. Дерево розбору тексту на семантичні токени

З малюнку 3.2. видно, що технологія NLP розкладає речення на токени. А за тим усі не смислові токени відхиляються й не беруться для подальшого аналізу. У даному прикладі було видалено визначник "an", який не несе ніякого смислового сенсу. А іменнику “Tom” було присвоєно особливий статус - “Власний іменник”, що означає найбільше семантичне навантаження в реченні. Відповідно він і буде мати найвищу позицію у списку вагової релевантності токенів.

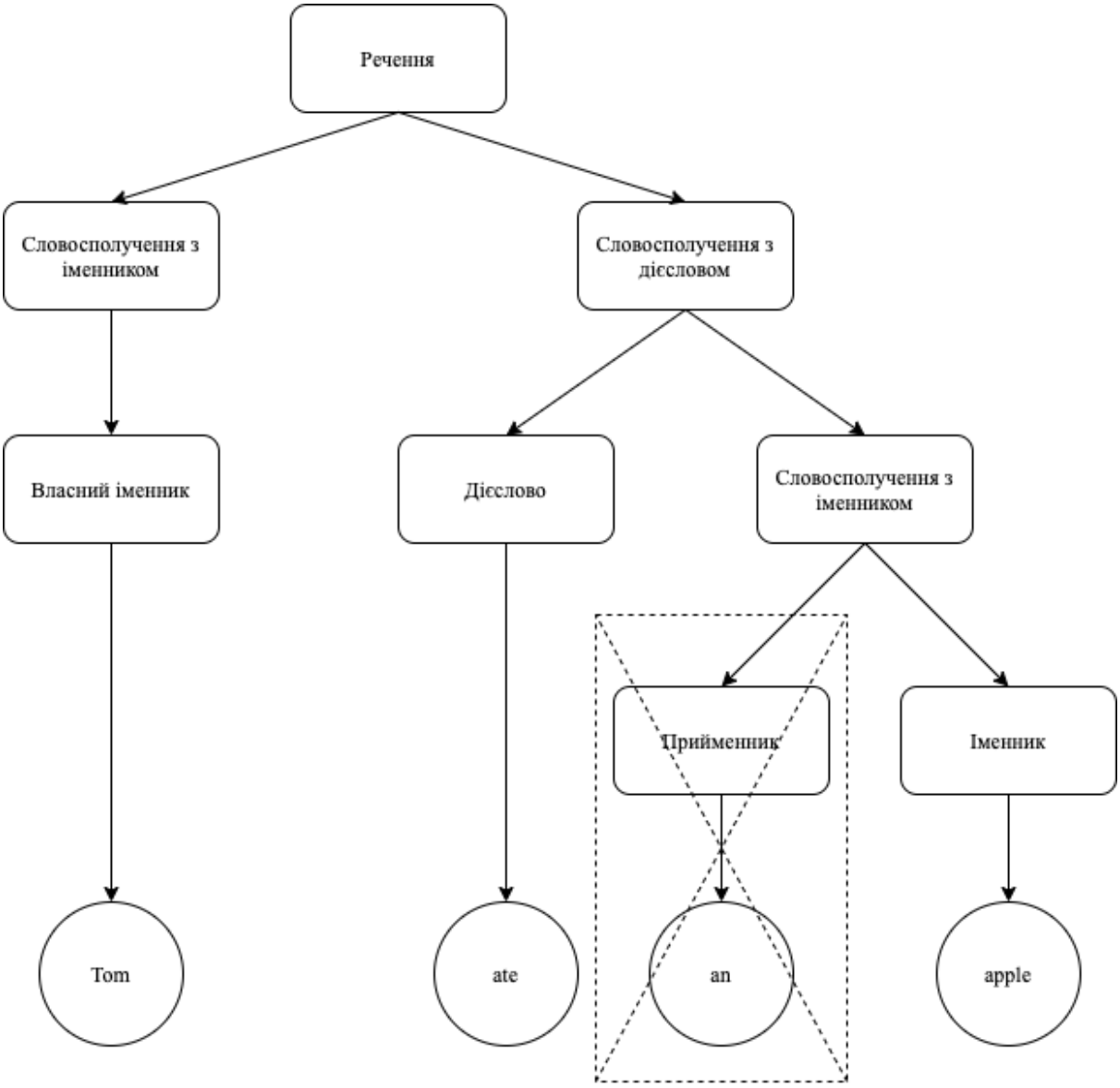

Рис. 3.2. Видалення семантично незначимих токенів

Усі сформовані токени записуються в базу даних до кожного запису. Тобто до кожного разу додається окреме поле із токенами заголовка фейкової статті, окремо поле із токенами заголовка статті спростування (якщо воно є). Також додатковим полем записуються токени тексту самого фейка та доведення спростування.

3.2.4. Порівняння токенів заголовку з використанням алгоритму Левенштейна

Додаток можна встановити як розширення веб-браузера. У такому випадку тексти для перевірки на правдивість обробляються автоматично, без додаткового

втручання користувача. Або користувач може надіслати запит на відкритий API чи завантажити новину на спеціальній веб сторінці розробленого програмного продукту.

Після того, коли API отримав повний контент новини, у який входять її заголовок, текст, теги, зображення, посилання, автор, веб-сайти, які поширюють цю інформацію. Проходить перевірна новини на правдивість.

На початковому етапі формується список токенів із заголовку вхідної статті за допомогою NLP. Потім відбувається пошук перевірених або спростованих фактів у базі даних з використанням обрахунку відстані Левенштейна.

В теорії інформації, лінгвістиці та інформатиці відстань Левенштейна або алгоритм Левенштейна - це рядкова метрика для вимірювання різниці між двома послідовностями. Неофіційно відстань Левенштейна між двома словами - це мінімальна кількість однозначних редагувань (вставок, вилучень чи підстановок), необхідних для трансформування одного слова в інше.

Відстань Левенштейна також може називатися дистанцією редагування, хоча цей термін може також позначати більшість метрик відстані, спільно відомих як відстань редагування. Це поняття тісно пов'язане з парним вирівнюванням рядків [16]. Приклад порівняння двох слів (токенів) показано на рисунку 3.3.

Програмна реалізація виглядає наступним чином. До уваги приймаються випадки, коли прослідковується збіг рядків. Мета алгоритма полягає в пошуку збігів для коротких фразах чи словах у набагато довших текстах, при цьому, слід очікувати невеликої кількості відмінностей. Наприклад, короткі фрази можуть походити зі словників. Де одна з фраз зазвичай коротка, а інша - довільно довга. Це широкий спектр застосувань, наприклад, перевірки орфографії, системи корекції оптичного розпізнавання символів та програмне забезпечення для перекладу на природну мову на основі перекладацької пам'яті.

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

Рис. 3.3. Обрахунок різниці Левенштейна між словом та словосполученням

Відстань Левенштейна можна також обчислити між двома більш довгими фразами чи рядками, але вартість його обчислення, яка приблизно пропорційна добутку двох довжин виразів, робить це непрактичним. Таким чином, коли їх використовують для нечіткого пошуку рядків у таких додатках, як зв'язок запису, порівняльні фрази зазвичай короткі (або використовуються просто слова), щоб поліпшити швидкість порівняння.

Також відстань Левенштейна застосовують як алгоритми нечіткого пошуку для виявлення однакових даних, що надходять до системи з різних джерел. Або в ролі програмного продукту для пошуку інформації за нечітким запитом.

У лінгвістиці відстань Левенштейна використовується як метрика для кількісного визначення мовної відстані або для визначення наскільки дві фрази відрізняються одна від одної. Вона пов'язана з взаємною розбірливістю, чим більше значення відстані, тим нижча взаємна розбірливість і чим менша значення

відстані, тим вища взаємна розбірливість. Тобто при меншому значенню обрахованої відстані слова чи фрази мають більше схожості в синтаксичному розумінні [17].

Отож, із заголовка вхідної статті формуються токени та порівнюються на основі різниці відстані Левенштейна із уже збереженими токенами фейкових статей. І чим меншою виявляється відстань, тим більша ймовірність, що дана стаття - фейкова.

Проте для означення статті як дезінформація, аналізу на основі порівняння відстані Левенштейна недостатньо.

По-перше, важливо враховувати скоринг по TF-IDF, тобто змістовну вагу слова в тексті, яке попало в різницю.

По-друге, для точного результату необхідно враховувати сам текст статті, а не тільки її заголовок. Також, можна посилатися на джерело інформації, адже в базі даних деякі з них мають позначку “фейковий ресурс”.

Скоринг по TF-IDF допомагає пришвидшити пошук схожих статей за рахунок оптимізації пошукових результатів у базі даних. Статистична міра для оцінки важливості слова в документі - скоринг TF-ID - повинна базуватися на схожості статей між собою. Чим меншого значення набуває цей показник - тим більш схожими мають бути статті.

Цей показник є прямо пропорційним до різниці Левенштейна між парою токенів та зворотно пропорційним до кількості слів у токені та ваги токена у тексті. Це дозволяє при пошуку надавати більшу вагу багатослівним та вагомим токенам статті, оскільки саме вони формують головну суть текста новини. При цьому, маленькі токени, які також можуть бути дуже важливими, не ігноруються, а теж враховуються при пошуку.

Запропонована модель скорингу дозволяє оптимізувати процес прийняття фінального рішення, оскільки грамотно формує перелік семантично схожих статей, які походять одна на одну не лише грамматично, а й змістовно, а це значно спрощує пошук спростування або ж підтвердження до контексту вхідної статті.

3.2.5. Порівняння токенів текстів статтей, у яких невелика різниця відстані Левенштейна в заголовку

Після порівняння заголовків статей, відбувається перевірка різниці відстані Левенштейна серед токенів текстів.

Для цього розроблено додатковий аналіз отриманих результатів. Та на його основі розглядається чотири можливих варіантів розвитку сценарію:

1) різниця відстані Левенштейна між вхідною новиною та записаними в базі правдивими статтями менше, ніж 10%. У такому випадку вхідна новина помічається як “правда” і записується до бази даних;

2) різниця відстані Левенштейна між вхідною новиною та збереженими в базі фейковими статтями менше, ніж 10%. У такому випадку вхідна новина помічається як “фейк” і записується до бази даних;

3) різниця відстані Левенштейна між вхідною новиною та записаними в базі правдивими статтями більше, ніж 90%. У такому випадку вхідна новина помічається як “фейк” і записується до бази даних;

4) різниця відстані Левенштейна між вхідною новиною та записаними в базі фейковими статтями більше, ніж 90%. У такому випадку вхідна новина помічається як “правда” і записується до бази даних;

У всіх інших випадках факт надсилається на ручну перевірку. Тобто журналісти-фактчекери проводять додаткове дослідження для перевірки новини. А потім записують її в базу даних як правдивий факт або як фейкову з обов’язковими полями “спростування” та “посилання на спростування”. Також допускається запис фейкової новини із окремими позначеннями як “маніпуляція” чи “часткова правда”, чи “джерело - жертва дезінформації”. Останнє поняття - це коли особа чи новинний ресурс розмістили фейкову інформацію на своїй сторінці, не підозрюючи, що це неправда.

3.2.6. Формування результатів програми автоматичного аналізу новин

Будь-яка оброблена вхідна новина буде записана в базу даних за результатами автоматичної перевірки або після додаткової пошукової роботи фактчекера.

Формування вибірки фактів відбувається наступним чином:

1) фільтрація фактів згідно токенів заголовків: у таку вибірку попадають факти, схожість токенів заголовків яких є більшою за 75% відповідно до відстані Левенштейна;

2) розподіл фактів на схожі та протилежні: відбувається порівняння токенів тексту. При схожості токенів більшій за 50%, факт маркується як схожий. У іншому випадку - присвоюється протилежний статус від базової статті, з якою було порівняння.

Результатом виконаних дій є перелік правдивих фактів, спростованих фейків, фейкових новин, відсортований за зменшенням релевантності до заданого списку токенів статей, із якими проводилося порівняння. Також треба врахувати окремий список статей,

3.2.7. Перевірка результатів

Для будь-якої автоматичної перевірки інформації необхідно виконувати перевірку людським фактором. Мається на увазі, що за всіма результатами автоматичної перевірки інформації, повинен слідкувати компетентний журналіст-фактчекер.

Результати програми, при яких різниці відстані Левенштейна змінюються в межах 10%, достатньо тільки відслідковувати. У інших випадках журналісту необхідно переглянути інформацію та переконатися в правильності роботи програми, наприклад, якщо відстань Левенштейна не більше 25%. Якщо ж результат показує від 25% до 50%, тоді фактчекери пророблюють перевірку за джерелом інформації та її коротким змістом і суттю. У всіх інших випадках фактчекеру необхідно обов’язково проводити додаткове журналістське розслідування щодо вхідної новини.

3.3. Пропозиції щодо подальшого розвитку програмного продукту

Запропонований спосіб розпізнавання фейкових новин продемонстрував доволі високу точність, але існує декілька шляхів для його покращення.

Наприклад, розпізнавати не тільки фейки, а й правдиві новини. Та маркувати їх як “правда” для засвідчення фактів у стрічці новин користувача.

Також даний спосіб можна застосувати для формування датасетів значно швидше та більших обсягів, ніж це можливо виключно людськими ресурсами. А потім застосувати сформовані датасети для навчання нейронної мережі.

Загалом до фейкових також відносяться такі типи новин як маніпуляція та часткова правда. Різницю між ними можна пояснити наступним чином.

З точки зору наміру, є дві основні категорії - “disinformation” та “misinformation”. Перше - свідоме введення в оману автором (але не конче - поширювачем), а друге - несвідоме (користувач поширює неправдиву новину навіть не підозрюючи, що це фейк). Таке поняття як часткова правда може бути і першим, і другим. А маніпуляція - лише першим.

Практично це можна перетворити на індекс маніпулятивності, який бере до уваги надійність джерела інформації. Якщо джерело надійне, то брак даних - misinformation, у іншому випадку - disinformation. Також ЗМІ можуть відображати не чисті факти, а їх інтерпретацію. Тому замість відображення дійсності вони створюють псевдореальність, у ній за допомогою стереотипів маркують та закріплюють необхідне сприйняття інформації. Тому розпізнавання усієї неправдивої інформації та класифікування її на фейки, маніпуляцію та часткову правду є не менш важливою проблемою.

Отже, одним із напрямків удосконалення може бути розпізнавання не тільки фейків, а й маніпуляції. А також відокремлення їх у базі даних за різною класифікацією.

3.4. Переваги та потенційні недоліки запропонованого способу аналізу новин

У нашому сьогоденні конкурентів у сфері автоматизованої перевірки новин на правдивість майже немає. Основний пласт роботи виконується виключно журналістами-фактчекерами, тому провести порівняльний аналіз запропонованого способу дуже складно. Проте, у ході розробки алгоритму виділено такі переваги:

- запропонований спосіб працює значно скоріше, ніж журналісти;
- показує більш ефективні результати, ніж фактчекери, адже повертає результат аналізу в актуальний для новини час (тобто в той самий день, коли з’явилася новина, а не тоді, коли через пару днів вона попала в архів);
- популяризує боротьбу із дезінформацією за рахунок цілодобової роботи онлайн, без зайвої комунікації користувача та журналіста;
- виступає як важливий діджитал інструмент в умовах інформаційної війни;
- спосіб як програмний продукт не має конкурентів на ІТ ринку, а отже має потенційно великий успіх для масової популяризації та стрімкого комерційного розвиток.

Проте у запропонованому рішенні поставленого завдання прослідковуються деякі недоліки:

- наповнення сформованої бази даних потребує постійного оновлення, тобто скрапінг статтей повинен запускатися кожні дві-три години, що значно навантажує систему;
- видані результати потребують постійної журналістської перевірки;

- запропонована модель включає поєднання декількох складних у реалізації модулів, а отже необхідна концентрація та кропітлива робота інженерів;
- реалізація запропонованого способу потребує значні ресурси на підтримку повноцінної мікросервісної архітектури.

Висновки до розділу 3

Третій розділ присвячений розробці алгоритму пошуку новин. У ньому описано та обумовлено вибір шляху наповнення бази даних правдивими новинами та фейкоовими разом із їх спростуваннями, застосування технології NLP для формування токенів, обрахування різниці токенів за алгоритмом Левенштейна, критерії оцінки новини на правдивість. Для побудови моделі алгоритму було запропоновано наступне:

1. Обгрунтований спосіб наповнення бази даних перевіреними новинами із надійних онлайн ресурсів, а також шлях оптимізації отриманого датасету.

2. Алгоритм токенізації вхідної статті. А саме, доцільність окремо розкладати заголовок та безпосередньо текст на токени, ефективність видалення слів, які не несуть семантичного навантаження в отриманому тексті, та запис токенів у базу даних разом зі статтями як окремі поля.

3. Спосіб порівняння токенів на схожість на основі обрахунку різниці відстані Левенштейна. При застосуванні цього способу для пошуку збігів у коротких фразах чи словах прослідковується знаходження невеликої кількості відмінностей за відносно короткий відрізок часу - декілька секунд. Хоча відстань Левенштейна можна також обчислити між двома більш довгими фразами чи рядками, але вартість такого обчислення, яка прямо пропорційна добутку двох довжин виразів, буде занадто довгою та не матиме практичного сенсу. Тому до уваги беруться тільки безпосередньо токени статті, а не її речення, абзаци чи сам текст.

4. Важливе доповнення розробленого алгоритму аналізу новин за рахунок врахування скоринг по TF-IDF - змістовну вагу слова в тексті, та перевірка посилатися на джерело інформації, адже в базі даних деякі з них мають позначку “фейковий ресурс”.

5. Пропозиції подальшого розвитку та вдосконалення запропонованого способу. А саме, розрізнення неправдивої інформації на три категорії: фейк, маніпуляція та часткова правда. А також врахування способу поширення

дезінформації: свідоме введення в оману користувача автором фейку або несвідоме (користувач поширює неправдиву новину навіть не підозрюючи, що це фейк).

Крім цього, було описано основні переваги та потенційні недоліки щодо застосування та розвитку запропонованого способу автоматичного аналізу фейкових новин.

РОЗДІЛ 4
РЕАЛІЗАЦІЯ АЛГОРИТМУ РОЗПІЗНАВАННЯ ФЕЙКОВИХ НОВИН

4.1. Засоби створення програмного продукту

Для реалізації запропонованого алгоритму було вирішено розробити три окремих програмних модулі: скрапер, пошук та REST API додаток для демонстрації можливостей алгоритму. Задача скрапера - періодично запитувати у джерела перевірених новин актуальні дані, аналізувати їх та зберігати результати аналізу. Задача пошуку - приймати статтю на вхід, аналізувати її, шукати подібні до неї статті у базі перевірених та на основі результату такого пошуку вирішувати, чи є дана стаття правдивою, або ж ні.

Для реалізації збереження та пошуку проаналізованих статей було обрано сервер ElasticSearch. Це пошукова система, що базується на бібліотеці Lucene. Вона надає розподілений та повнотекстовий пошуковий механізм, із веб-інтерфейсом HTTP та документами JSON без схем. Elasticsearch розроблений на Java. Після відкритої бізнес-моделі частини програмного забезпечення ліцензуються під різними ліцензіями з відкритим кодом (переважно ліцензія Apache), а інші частини підпадають під власну (доступну для джерела) ліцензію Elastic. Офіційні клієнти доступні на Java, .NET (C #), PHP, Python, Apache Groovy, Ruby та багатьох інших мовах. Згідно з рейтингом DB-Engines, найпопулярнішою пошуковою системою підприємств є ElasticSearch, за якою слідує Apache Solr, також заснована на Lucene. ElasticSearch можна використовувати для пошуку всіх видів документів. Він забезпечує масштабований пошук, здійснює пошук у режимі реального часу та підтримує багатосторонність. ElasticSearch є розподіленою системою, що означає, що індекси можна розділити на фрагменти, і кожен може мати різну кількість реплік. Кожен вузол розміщує один чи більше контейнерів і виконує функцію координатора для делегування операцій правильним фрагментам. Маршрутизація здійснюється автоматично. Пов'язані дані часто зберігаються в

одному індексі, який складається з одного або декількох первинних контейнерів і декількох реплік. Після створення індексу кількість первинних контейнерів неможливо змінити [18].

Для реалізації скрапінгу було обрано бібліотеку BeautifulSoup 4. Це бібліотека, написана на мові програмування Python для розбору HTML і XML-документів (у тому числі з неправильною розміткою, тобто незакритими тегами). Вона створює дерево розбору для сторінок, яке може бути використано для отримання даних з HTML, що корисно для зчитування вмісту веб-сторінок [19].

Для реалізації REST API додатку для демонстрації алгоритму було обрано мову програмування Python та бібліотеку Django. Django - це безкоштовний веб-фреймворк, заснований на Python, який відповідає архітектурній схемі модельного шаблону (MTV). Він підтримується Фондом програмного забезпечення Django (DSF), незалежною організацією, створеною як некомерційна організація 501.

Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Фреймворк наголошує на повторному використанні та «підключеності» компонентів, меншій кількості коду, низькому з'єднанні, швидкому розвитку та принципі не повторювати себе. Python використовується у всьому циклі розробки, навіть для файлів налаштувань та моделей даних. Django також надає додатковий адміністративний інтерфейс для створення, читання, оновлення та видалення, який динамічно генерується за допомогою самоаналізу та налаштовується через адміністраторські моделі.

Незважаючи на наявність власної номенклатури, як-от іменування об'єктів, що викликаються, генеруючи відповіді HTTP, основна структура Django може розглядатися як архітектура MVC. Фреймворк складається з об'єктно-реляційної моделі (ORM), який здійснює посередництво між моделями даних (визначеними класами Python) та базою даних ("Модель"), системою обробки HTTP-запитів із системою веб-шаблонів ("Вид") та диспетчером URL-адрес на основі регулярних виразів ("Контролер").

Система конфігурації Django дозволяє сторонньому коду підключатись до звичайного проекту за умови, що він дотримується конвенцій програми багаторазового використання. Більше 2500 розширень доступні для доповнення оригінальної поведінки фреймворка, надаючи рішення для задач, якими не переймається сам фреймворк: реєстрація, пошук, скрапінг, API інтерфейс, CMS тощо.

Однак ця розширюваність пом'якшується залежностями внутрішніх компонентів. Хоча філософія Django передбачає слабке з'єднання, фільтри шаблонів і теги передбачають одну реалізацію двигуна, і як пакети програм Auth, так і адміністратори вимагають використання внутрішнього ORM. Жоден із цих фільтрів або пакетних програм не є обов'язковим для запуску проекту Django, але багаторазові програми, як правило, залежать від них, спонукаючи розробників продовжувати використовувати офіційний стек, щоб повною мірою скористатися екосистемою додатків.

Django використовується багатьма відомими сервісами, серед яких варто визначити Службу громадського мовлення США, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket та Nextdoor.

4.2. Структура програмного продукту

В процесі розробки було виокремлено два етапи: реалізація алгоритму аналізу тексту та використання алгоритму під час скрапінгу та пошуку. Під час розробки програмного продукту були поставлені вимоги розробити гнучке рішення, яке буде мати багато модулів, що дозволить з часом змінювати певні компоненти системи, при цьому не змінюючи всю систему. Таким чином, в результаті аналізу, було розділено весь програмний продукт на наступні окремі модулі:

- Модуль текстового аналізу (FNTextAnalysis)
- Модуль скрапінгу (FNScrapper)
- Модуль пошуку (FNSearch)
- Модуль прийняття рішень (FNDecider)

- Модуль серверу для демонстрації (FNServer)

Нижче розглянуто більш детальний опис реалізації кожного з модулів.

4.2.1. Модуль текстового аналізу

Модуль текстового аналізу FNTextAnalysis є незалежним модулем, який представляє собою бібліотеку, написану на мові програмування Python, що імпортується у інші модулі за допомогою менеджера бібліотек pip. Головна задача модулю - виконувати текстовий аналіз вхідної статі та формувати список змістовних токенів цієї статті.

Відповідно до задач, модуль містить 3 головні класи:

- HeaderAnalyzer - аналізує заголовок статті
- BodyAnalyzer - аналізує текст статті
- CoreAnalyzer - містить спільний функціонал аналізу для HeaderAnalyzer та BodyAnalyzer

Безпосередній аналіз тексту відбувається у декілька етапів:

- видалення службових слів
- синонімайзинг токенів
- зведення токенів до нормальної форми
- формування списку семантичних токенів
- зведення списку до множини

Під час видалення службових слів використовуються stop token filter та lowercase token filter. Після роботи цих фільтрів з тексту видаляються всі службові слова та всі великі букви замінюються на малі. Службові слова не несуть ніякого семантичного навантаження на текст, а зведення всіх букв до малих спрощує подальший аналіз тексту.

Під час синонімайзингу за допомогою заздалегідь сформованого словника відбувається заміна синонімів на їх оригінальні слова. Наприклад, якщо в словнику міститься запис “анімації - анімація”, то після виконання цього етапу всі токени “анімації” буде замінено на токени “анімація”. Такі словники зазвичай містять заміни різних форм одних і тих самих слів, які не несуть у собі

додаткового семантичного навантаження. За рахунок цього зменшується навантаження на подальший аналіз токенів.

Наступним етапом є зведення токенів до нормальної форми. Під час цього етапу більшість токенів переводиться у просту форму, що значно полегшує подальший аналіз.

Після цього формується список семантичних токенів. Цей список містить у собі токени від одного до трьох слів, які вже пройшли всі попередні етапи. Саме на основі таких токенів і відбувається фінальний аналіз тексту.

Фінальним етапом є зведення токенів до множини (set). Під час цього видаляються всі дублікати, але підраховується їх кількість. Тобто кожен токін має додаткове поле в записі із кількістю його повторів у тексті.

Таким чином, після всіх етапів аналізу система повертає асоціативний масив токенів та кількості згадувань у тексті, який відсортований за кількістю, від великої до малої, та список токенів за тим порядком, у якому вони розташовані у тексті. Після цього відбувається опрацювання цих даних, в результаті якого виділяється масив найбільш семантично важливих токенів. Саме цей масив і є фінальним результатом аналізу тексту.

4.2.2. Модуль скрапінгу

Модуль скрапінгу FNScraper є незалежною бібліотекою, написаною на мові програмування Python. Від імпортує в себе модуль текстового аналізу FNTextAnalysis. Головна задача цього модулю - періодично завантажувати статті з перевірених джерел, аналізувати їх, видаляти дублікати та індексувати у ElasticSearch.

Нижче коротко описано зміст основних класів модуля скрапінгу FNScraper:

- Fetcher - завантажує перелік статей з перевіреного джерела;
- Analyzer - використовує модуль FNTextAnalysis для того, щоб проаналізувати статтю та робить попередню підготовку даних для зберігання у базу даних;

- Indexer - індексує отримані дані в ElasticSearch після завершення аналізу;

4.2.3. Модуль пошуку

Модуль пошуку FNSearch є незалежним модулем, написаним на мові програмування Python. Головна його задача - шукати статті у індексі ElasticSearch за набором змістовних токенів, отриманих під час аналізу та виконувати ранжування статей таким чином, щоб у результаті вони були відсортовані за схожістю за змістом.

До складу розробленого модуля пошуку входять нижче описані класи:

- QueryBuilder - будує фільтруючий запит в базу;
- ScoreBuilder - розраховує формулу розрахунку показника схожості;
- Engine - формує повний запит до ElasticSearch, використовуючи дані QueryBuilder та ScoreBuilder

Пошук та ранжування відбуваються одночасно, оскільки ElasticSearch має вбудовану модель оцінки документа. Це дозволяє під час пошуку обраховувати певний показник, за яким потім ці документи будуть сортуватися. У даному випадку до обрахування цього показника застосовується алгоритм Левенштейна, який дозволяє порахувати різницю між двома токенами. Відповідно до результатів підрахунку, чим меншою є ця різниця, тим більш схожими є токени.

Загальна модель побудови сортування документів формується на основі обрахунку показника схожості. Цей показник вираховується як сума значень алгоритму Левенштейна між кожною парою токенів, при цьому кожне з цих значень ділиться на кількість слів у токені та на коефіцієнт семантичної важливості токена (TF-IDF). Формула 2 відображає загальний обрахунок показника схожості:

$$SimilarityIndicator = \sum_0^i \frac{L_i}{n_i \times TF\text{-}IDF_i} \qquad (2),$$

де L_i - показник різниці відстані Левенштейна між i-ми токенами двох текстів, n_i - кількість слів у i-му токені, $TF\text{-}IDF_i$ - коефіцієнт семантичної важливості i-го токена.

Отримані значення сортуються відповідно до показника схожості від меншого до більшого. Відповідно, чим меншим є значення цього показника - тим більш схожою є стаття з пошукової видачі до вхідної статті.

4.2.4. Модуль прийняття рішень

Модуль прийняття рішень FNDecider є незалежною бібліотекою, написаною на мові програмування Python. Його задача - отримувати проаналізовану статтю, пошуковий результат зі схожими статтями з бази даних та на основі цих даних приймати рішення: чи є вхідна стаття правдивою, чи неправдивою, чи потребує додаткового журналістського дослідження.

Алгоритм перевірки полягає в пошуку максимально близьких за змістом статей у пошуковій видачі ElasticSearch. Розглядаються наступні сценарії:

- у видачі результатів пошуку ElasticSearch знайдена правдива стаття, яка схожа з аналізованою на більше, ніж 90%. При такому сценарію аналізована стаття помічається як “правда” і записується до бази даних;
- у результаті пошуку ElasticSearch виявлено фейкову статтю, яка схожа з аналізованою на 90% і більше. У такому випадку аналізована стаття помічається як “фейк” і записується до бази даних, а посилання на спростування додається з уже існуючого запису;
- у видачі результатів пошуку ElasticSearch знайдена правдива стаття, яка протилежна з аналізованою на 90% і більше. У такому випадку аналізована стаття помічається як “фейк” і записується до бази даних;
- у видачі результатів пошуку ElasticSearch знайдено фейкову статтю, яка протилежна з аналізованою на більше, ніж 90%. При цьому сценарію аналізована стаття помічається як “правда” і записується до бази даних;

Якщо за наведеними вище критеріями не знайдено жодної статті у базі даних - аналізована стаття маркується як така, що потребує додаткового журналістського розслідування. Однаковість статей перевіряється завдяки показнику схожості, який обраховуєтьсчя під час пошуку статей у базі даних.

4.2.5. Модуль серверу для демонстрації

Модуль серверу для демонстрації можливостей алгоритму FNServer реалізований за допомогою мови програмування Python та фреймворку Django. Задача модулю - надати програмний інтерфейс для користування реалізованими модулями пошуку та скрапінгу.

Модуль реалізований у вигляді серверу з набором API методів. Розглянемо їх:

- POST /start_scrapping - починає процес скрапінгу
- POST /stop_scrapping - зупиняє процес скрапінгу
- POST /analyze - аналізує задану новину

Методи не потребують авторизації та працюють за асинхронною моделлю, а саме: валідують запит та відправляють його на виконання, відправляють клієнту повідомлення про успішну обробку запита. Виключення - запит на аналіз новини, який працює синхронно, тобто сервер отримує запит, обробляє його та формує відповідь у одному потоці. Тобто клієнт замість повідомлення про очікування - “Ваш запит прийнятий на обробку.” - відразу отримує результати аналізу автоматичного фактчекінгу.

4.3. Експериментальні дані та результати

Над розробленим програмним продуктом було проведено тестування, яке складалося з наступних етапів:

- скрапінг перевірених новин;
- аналіз вибірки новин;
- перевірка результатів людським фактором.

Для тестування використовується модуль FNServer. Сервер запускається на основі локального середовища (localhost) та з використанням порту 8181. Запуск та зупинка скрапінгу зображені на рис. 4.1.

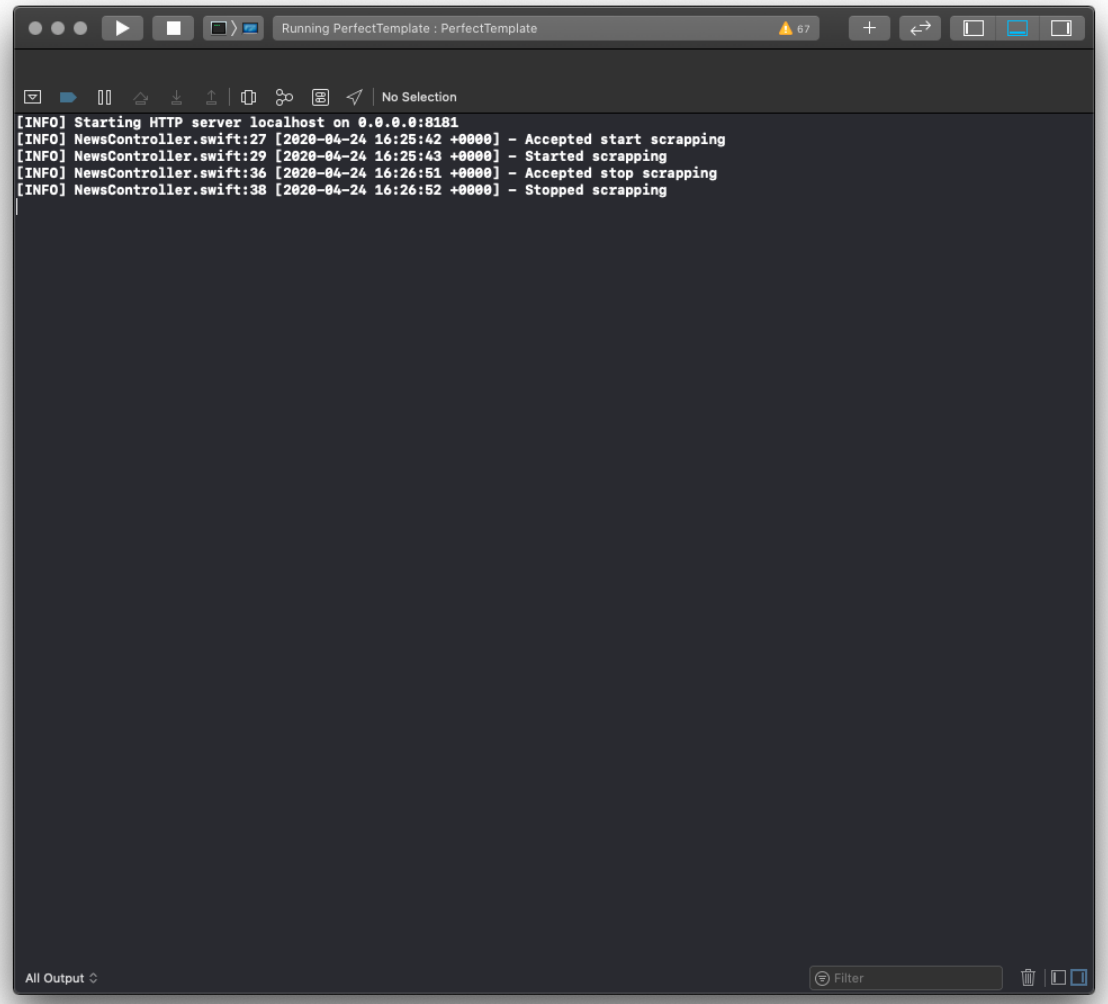

Рис. 4.1. Запуск та зупинка скрапінгу

Для тестування запропонованого вище способу обрано новини з групи сайтів sputniknews за період із 1.02.2020 по 29.02.2020. Для формування бази даних із спростуваннями було обрано дані з сайту euvsdisinfo.eu. Приклад новини на сайті euvsdisinfo.eu зображено на рисунку 4.2.

Після закінчення скрапінгу база даних спростованих фейків містила 7837 записів, а із сайту southfront.org було записано 500 новин. Скрапінг тривав 3 години, в той час як перевірка вибірки новин тривала 12 хвилин.

У середньому аналіз однієї новини триває приблизно 1,2 секунди, за одну хвилину аналізується приблизно 43 новини.

Рис. 4.2. Приклад фейкової новини на сайті euvsdisinfo.eu

Під час перевірки новини відбувається аналіз отриманого тексту з використанням модулю FNTextAnalysis, формування токенів та пошук новин за токенами у базі з використанням модулю FNSearch та прийняття рішення з використанням модулю FNDecider. Запит є синхронним, триває близько 1,2 секунди (тривалість лінійно залежить від розміру тексту новини), результатом запиту є статус новини - truth, якщо новина є правдивою і відповідно false, якщо новина є неправдивою. Приклад такого запиту зображений на рисунку 4.3.

Результати аналізу новин запропонованим способом наведено в таблиці 4.1. Як показано в таблиці результатів, проаналізовано 500 новин. 79 із них виявилися фейковими, 45 - відправлено на додаткове опрацювання фактчекерами. Для всіх інших новини спростування не було знайдено в базі даних, отже вважається, що вони правдиві та записуються в базу даних із позначенням “правда”. Проте, такі статті потребують додаткової перевірки фактчекером або компетентного журналістського розслідування.

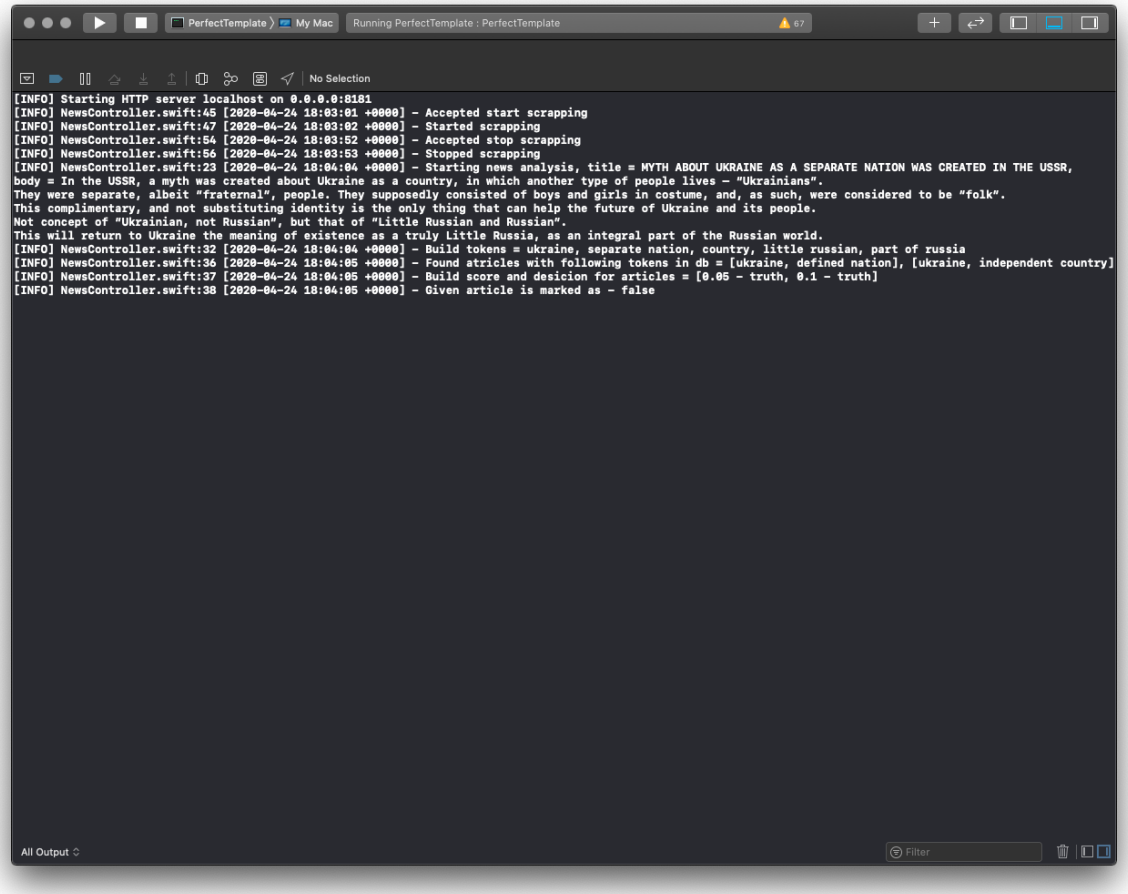

Рис 4.3. Запит на аналіз новини

Таблиця 4.1.

Результати виконання роботи розробленого способу

Категорія	Кількість
Фейки	79
Додатково опрацьовані	45
Правда	376

Для перевірки правильності роботи алгоритму було застосовано класичний журналістський метод перевірки новини. Він включає аналіз новин у перевірених джерелах та співставлення їх з даною статтею для прийняття

фінального рішення. Проаналізувавши всі новини журналістським методом, результати були наступними (показано в таблиці 4.2.).

Таблиця 4.2.

Результати перевірки новин
на основі журналістського дослідження

Категорія	Кількість
Фейки	95
Правда	405

Таким чином, програма змогла правильно розпізнати 83% фейків. З огляду на те, що журналістські розслідування однієї статті можуть займати віл години часу до декілька діб, у той час, коли алгоритму потрібно всього трохи більше секунди, результат можна вважати позитивним. А навіть і більше, оскільки задача виявлення фейків є надважливою в сьогоденні, тому такі результати можуть допомогти журналістам доволі сильно збільшити свою продуктивність, оскільки зменшують кількість ручної роботи на 83% і цей показник може збільшуватися прямо пропорційно зі зростанням кількості записів у базі даних.

Висновки до розділу 4

У рамках запропонованого розділу розглянуто реалізацію програмного продукту для розпізнавання фейкових новин у мережі Інтернет, обґрунтовано вибір мови програмування та технології, які необхідні для вирішення поставленої задачі. Окрім цього, у четвертому розділі було вирішено наступні завдання:

1. Обґрунтовано вибір мови програмування Python, технологій NLP, TF-IDF, бібліотек BeautifulSoup, re, фреймворку Django нереляційної бази даних з технологією швидкого пошуку ElasticSearch та алгоритм обрахунку відстані Левенштейна для розробки програмних продуктів. Наведено їх короткий огляд, перераховано переваги кожної з них. Вказано головні парадигми, які використовувались під час розробки проекта.
2. Проведено детальний опис компонентів програми, а саме - модулі, які відповідають за текстовий аналіз, скрапінг, пошук, прийняття рішень, роботу серверів для демонстрації результатів. Перераховано основні функції, що забезпечують вказані модулі, а також способи їх реалізації та вимоги до їх роботи.
3. Описано процес розробки алгоритмів збору даних, токенізації текстів на основі Natural Language Processing, обрахування різниці відстані Левенштейна між токенами різних текстів, обчислення показника схожості TF-IDF.
4. Наведено сценарій виконання основних компонентів програми, вказано покрокові коментарі для взаємодії користувача із графічним інтерфейсом.
5. Проведено тестування способу, який опрацював вибірку з 500 новин. Для проведення аналізу запропонованого способу наповнено базу даних перевіреними новинами із сайту euvsdisinfo.eu. Для тестування цього способу обрано групу сайтів sputniknews (Sputnik Greece, Sputnik Polska, Sputnik Armenia та інші), які зазвичай поширюють фейкові новини. Результати тестування порівняно із результатами журналістської бази даних фейкових новин. Спосіб

правильно розпізнав 83% сфальсифікованих фактів. При чому, на опрацювання однієї статті витрачено від 1,7 с до 57 с часу. У той час коли фактчекери витрачають на перевірку однієї статті від трьох годин до декількох діб, що вагомо показує різницю у швидкості отримання результатів. Таким чином запропонований спосіб показав себе як ефективний для розпізнавання фейкових новин із доволі високою точністю.

РОЗДІЛ 5
РОЗРОБКА СТАРТАП ПРОЕКТУ

5.1. Короткий опис проекту

Проект має на меті перевірку контенту постів користувачів на фейковість. А також поєднує якісний контент та його поширення користувачами соціальних мереж.

Додаток повинен перевіряти новину, опубліковану користувачем або його друзями у соціальній мережі, на фейковість та надіслати зворотню відповідь на пошту користувача. Для залучення останніх активно застосовувати додаток, використовується гейміфікація, де кожен може штурмувати фейкарів «вогнем».

Перевірка текстів юзерів виконується шляхом порівняння їх з якісним контентом, який раніше зібраний у базу даних. Наповнюється база даних тільки якісним контентом від професійних проектів перевірки фактів. Тобто з сайтів, видань, блогів професійних журналістів-фактчекерів.

5.2. Бізнес-модель

Бізнес-модель FakesRadar близька до моделі "Захоплення чистого простору" Марка Джонсона.

5.2.1. Цінність продукту

Цінність продукту полягає в автоматизації перевірки стрічки новин на наявність дезінформації. Також важливим є здатність нашого сервісу повернути опис реальної ситуації щодо якої була створена дезінформація.

Для постачальників контенту сервіс важливий тим, що дозволяє надіслати інформацію користувачам соцмереж через їх друзів. Це гарантує підвищену увагу користувачів соцмереж до отриманої інформації.

Виконує вимоги: Сервіс допомагає користувачам виконати суспільно-важливу функцію боротьби проти дезінформації та відчути себе учасником важливої та корисної справи.

5.2.2. Сегмент споживачів

Проект належить до типу сервісів з двостороннім ринком. Першою аудиторією проекту є суспільно активні користувачі соціальних мереж. Другою аудиторію проекту є сервіси перевірки фактів, а також постачальники іншого якісного контенту, такі як видавництва, університети, онлайн-курси тощо.

5.2.3. Канали збуту

Основним каналом збуту для першої аудиторії є соціальні мережі. Каналом збуту для другої аудиторії є форуми, видання, зустрічі та інші професійні об'єднання.

5.2.4. Взаємодія зі споживачами

Залучення: Залучення споживачів з першої аудиторії відбуватиметься через їх друзів у соціальних мережах. Залучення споживачів другої аудиторії буде відбуватися через прямі контакти та професійні мережі.

Підтримка: Після комерційного запуску проекту передплатники сервісу матимуть цілодобову сервісну підтримку.

5.2.5. Дохід (монетизація)

Монетизація відбуватиметься за моделлю близькою до монетизації антивірусних сервісів. Додатковим джерелом доходів будуть тематичні проекти з постачальниками контенту та проектів протидії фальсифікації виборів.

5.2.6. Ціннісна пропозиція

Сервіс дозволяє користувачам перевірити свої соціальні мережі на наявність фейків та повідомити друзів про те, що вони стали жертвами дезінформації. Оскільки проект отримує перевірені дані від професійних проектів перевірки фактів, користувачі також можуть надіслати своїм друзям лінки на спростування дезінформації. Проектам фактчекерів ми допомагаємо поширювати результати їх роботи.

5.2.7. Ключові ресурси та ключові процеси

Основними процесами проекту є збір даних від фактчекерів та поширення даних через користувачами завдяки гейміфікації.

Матеріальні:

- забезпечення комп'ютерами усіх членів команди;
- використовуватимуться сервери для швидкої обробки великих даних;
- бази даних для зберігання правильного контенту.

Інтелектуальні:

- власні технічні розробки, розрахунки, комп'ютерн програми;
- власні розробки machine learning, AI, elastic search.

Людські:

- Developer-server
- Developer-Android
- Developer-iOS
- Marketer
- Accountant
- CEO
- CFO
- CTO
- CPO
- Designer
- Content Manager

Формула отримання прибутку:

Передплата на сервіс (на кшталт антивіруса), контракти з власниками контенту.

5.2.8. Ключові партнери

Ключовими партнерами проекту є Центр демократії, розвитку та верховенство права Стенфордського університету та міжнародний акселератор Startup Wise Guys, одна із найуспішніших естонських компаній RebelRoam.

5.3. Дослідження конкурентного оточення

Схожим проектом на ринку є проект Fake News Guard. Втім, нема сенсу вважати його прямим конкурентом через відсутню гейміфікацію та можливість прямого застосування до стрічки новин у соціальних мережах.

5.4. Маркетингова стратегія просування

Перевірка сервісом стрічки новин завжди буде безкоштовною якщо здійснюватиметься через розроблений сайт. Основним методом просування буде досягнення довіри у користувачів і рекомендації їх своїм друзям в соцмережах. Додатковим методом просування буде реклама в соціальних мережах можливості протидії дезінформації тут і зараз кожним користувачем.

5.5. Резюме

У наш час дуже важко знайти людину, яка б не чула про соціальні мережі хоч щось. Щоденно ми заходимо в «Фейсбук» чи «Інстаграм», щоб погортати стрічку новин чи перевірити нові лайки на сторінці. Але чи знаємо ми що правда в нашій стрічці, а що міф?

Здавалось би, це ж публікують наші друзі, які у реальному житті не будуть нас обманювати чи говорити різну нісенітницю. Але, у віртуальному, на жаль, усе по-іншому. Більше 90% користувачів соціальних мереж не перевіряють інформацію з декількох джерел, а репостять її довірившись одному. А це, зазвичай, велика помилка! Адже, у буремні часи інформаційної війни фейкарі так і чекають легкої довіри.

Таким чином, додаток FakesRadar допоможе кожному розпізнати фейкові новини від реальних у стрічці новин у будь-якій соціальній мережі. І що дуже

важливо, дозволить раціонально використовувати час, адже результат користувач побачить уже за декілька секунд, що вкрай необхідно у сучасному ритмі життя.

5.6. Подальші кроки в проекті

5.6.1. Наукова діяльність

Заплановано дослідження про взаємодію виробників дезінформації між собою, про взаємодію проектів перевірки фактів, і, найголовніше, дій користувачів щодо визначення дезінформації та її ліквідації у своїх соцмережах.

5.6.2. Організаційна діяльність

Проект починається як некомерційний проект, який планує надалі надавати послуги і продавати їх. Прибутки від продажів послуг будуть спрямовані на розвиток проекту.

5.6.3. Маркетингова діяльність

Поступово буде розширено теми просвітницьких кампанії, регіони роботи сервісу та мови контенту.

5.6.4. Комерційна діяльність

Вважається готовність користувачів та клієнтів платити за надані послуги розробленим програмним продуктом одним з основних критеріїв успішності роботи, каналом взаємодії з аудиторіями та механізмом регуляції розвитку проекту. Тому, приділяється велике значення розвитку здатності розробленого сервісу надавати комерційні послуги.

Висновки до розділу 5

П’ятий розділ містить контент по розробці стартап-проекту на основі запропонованого програмного продукту, зроблено опис ідеї проекту, бізнес-модель, маркетингову ціль, проведено детальний аналіз конкурентів та можливостей, що доступні на сьогоднішньому ринку. Для дослідження в рамках розробки стартап-проекту було виконано наступні завдання:

1. Наведено загальний опис ідеї проекту для виходу на IT-ринок, описано основне призначення розробки.
2. Розроблено бізнес-модель запропонованого способу, вказано цінність продукту, сегмент споживачів, канали збуту. Розраховано спосіб монетизації проекта, його ціннісну пропозицію.
3. Вказано ключові ресурси та ключові процеси, необхідні для розвитку та нормального існування програмного продукту.
4. Досліджено конкурентне оточення з точки зору програмної реалізації ідеї та ключові партнери для інвестування та підтримки маркетингової стратегії. На основі цього дослідження розроблено фінансовий план
5. Описано подальші кроки в проекті, зокрема наукова, організаційна, маркетингова та комерційна діяльності.

Як результат, розроблено стартап-проект для запропонованого способу, який готовий до виходу на IT ринок, отримано навички створення стартап-проектів, побудови маркетингової стратегії та аналізу обраного ринку.

ВИСНОВКИ

Магістерська робота присвячена вирішенню задачі пошуку фейкових новин у мережі Інтернет.

У першому розділі цієї роботи було проведено теоретичний огляд та аналіз особливостей технології Natural Language Processing, алгоритму Левенштейна та скорингу TF-IDF, виявлено основні переваги та недоліки їх застосування. Проаналізовано відомі рішення автоматичного спростування фейкової інформації. Проведено дослідження сучасного стану розвитку технології NLP у мобільних застосунках, розглянуто основні структурні елементи Natural language processing та проблеми, які дозволяють повністю або частково вирішення впровадження цієї технології.

У рамках другого розділу проведено дослідження інструментів токенізації текста за реченнями та словами. Показано доречність застосування деяких так званих “корпусів” бібліотеки NLTK, а саме Brown Corpus, Reuters Corpus, Inaugural Address Corpus, Parsed Corpora та інші. Проаналізовано та доведено ефективність токенізації, лематизаї та стемінгу текстів. Що застосовується для поділу їх на слова чи словосполучення, а також приведення слів до нормальної словникової форми. Аргументовано необхідність обрахунку значення показника схожості TF-IDF, який є статистичною мірою для оцінки важливості слова в документі.

Третій розділ присвячений розробці алгоритму пошуку новин. У цьому розділі запропоновано спосіб наповнення бази даних перевіреними новинами із надійних джерел, а також шлях оптимізації отриманого датасету. Розроблено алгоритм токенізації вхідної статті. Доведено доцільність окремо розкладати заголовок та безпосередньо текст на токени, ефективність видалення слів, які не несуть семантичного навантаження в отриманому тексті, та запис токенів у базу даних разом зі статтями як окремі поля. Доведено важливість доповнення розробленого алгоритму аналізу новин за рахунок врахування скоринг по TF-IDF - змістовну вагу слова в тексті, та перевірка посилатися на джерело інформації, адже у базі даних деякі із них мають позначення “фейковий ресурс”. Окрім

цього, були описані основні переваги та потенційні недоліки в розробці, технічній підтримці та застосуванні запропонованого способу аналізу фейкових новин.

У рамках четвертого розділу розглянуто реалізацію програмного продукту для розпізнавання фейкових новин у мережі Інтернет. Обґрунтовано вибір мови програмування Python, технологій NLP, TF-IDF, бібліотек BeautifulSoup, re та фреймворк Django для розробки програмних продуктів. Наведено їх короткий огляд, перераховано переваги кожної з них. Проведено детальний опис компонентів програми, а саме - модулі, які відповідають за текстовий аналіз, скрапінг, пошук, прийняття рішень, роботу серверів для демонстрації результатів. Перераховано основні функції, що забезпечують вказані модулі, а також способи їх реалізації. Описано процес розробки алгоритмів збору даних, токенізації текстів на основі Natural Language Processing, обрахування різниці відстані Левенштейна між токенами різних текстів, обчислення показника схожості TF-IDF.

Для перевірки запропонованого рішення було проведено тестування способу, який опрацював вибірку з 500 новин. Для проведення аналізу запропонованого способу наповнено базу даних перевіреними новинами із сайту euvsdisinfo.eu. Для тестування цього способу обрано групу сайтів sputniknews (Sputnik Greece, Sputnik Polska, Sputnik Armenia та інші), які зазвичай поширюють фейкові новини. Результати тестування порівняно із результатами журналістської бази даних фейкових новин.

Спосіб правильно розпізнав 83% сфальсифікованих фактів. При чому, на опрацювання однієї статті витрачено від 1,7 с до 57 с часу. У той час, коли фактчекери витрачають на перевірку однієї статті від трьох годин до декількох діб, що показує вагому різницю у швидкості отримання результатів. До того ж, актуальність новини триває не більше однієї – двох діб. Тобто іноді результати журналістського розслідування надходять пізніше, ніж новина потрапляє в архівні дані.

Таким чином, можна вважати, що поставлену ціль – підвищення ефективності системи виявлення, синтезу та аналізу фейкових новин шляхом застосування технології Natural Language Processing – досягнуто. Запропонований спосіб реалізації поставленого завдання показав себе як ефективний для розпізнавання фейкових новин із доволі високою точністю та виявився значно швидшим від існуючих варіантів рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fake news guard [Електронний ресурс] - режим доступу до видання: https://www.fakenewsguard.com

2. Snopes [Електронний ресурс] - режим доступу до видання: https://www.snopes.com/about-snopes/

3. Natural Language Toolkit [Електронний ресурс] - режим доступу до документації: https://www.nltk.org

4. sentence tokenization.py [Електронний ресурс] - режим доступу до ресурсу: https://gist.github.com/Ventsislav-Yordanov/39237759c087ac4151b3c06d4e566747

5. Основы Natural Language Processing для текста [Електронний ресурс] - режим доступу до ресурсу: https://habr.com/en/company/Voximplant/blog/446738/

6. NLP Pipeline: Stop words [Електронний ресурс] - режим доступу до ресурсу: https://medium.com/@makcedward/nlp-pipeline-stop-words-part-5-d6770df8a936

7. Corpus Readers документація http://www.nltk.org/howto/corpus.html

8. Accessing Text Corpora and Lexical Resources [Електронний ресурс - режим доступу до документації: http://www.nltk.org/book/ch02.html

9. Регулярні вирази [Електронний ресурс] - режим доступу до наукового онлайн видання: https://echo.lviv.ua/dev/740

10. Метасимволи в регулярних виразах [Електронний ресурс] - режим доступу до онлайн документації: https://support.google.com/analytics/answer/1034324?hl=uk

11. Re — Regular expression operations [Електронний ресурс] - режим доступу до ресурсу: https://docs.python.org/3/library/re.html

12. TF-IDF from scratch in python on real world dataset [Електронний ресурс] - режим доступу до журналу: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

13. EUvsDisinfo [Електронний ресурс] - режим доступу до ресурсу: https://euvsdisinfo.eu/about/

14. StopFake [Електронний ресурс] - режим доступу до ресурсу: https://www.stopfake.org/uk/pro-nas/

15. Natural Language Toolkit (NLTK) [Електронний ресурс] - режим доступу до ресурсу: https://www.nltk.org/book/ch00.html

16. *The Levenshtein-Algorithm* [Електронний ресурс] - режим доступу до ресурсу: http://www.levenshtein.net

17. Levenshtein distance [Електронний ресурс] - режим доступу до ресурсу: https://en.wikipedia.org/wiki/Levenshtein_distance#cite_note-navarro-2

18. ElasticSearch [Електронний ресурс] - режим доступу до ресурсу: https://www.elastic.co

19. BeautifulSoup [Електронний ресурс] - режим доступу до ресурсу: https://www.crummy.com/software/BeautifulSoup/#Download

20. Bird Steven. *Natural Language Processing with Python* / Bird Steven, Edward Loper and Ewan Klein: O'Reilly Media Inc, 2009 - ст. 87 - 93.

21. Nltk.corpus treebank [Електронний ресурс] - режим доступу до ресурсу: https://books.google.com.ua/books?id=KGIbfiiP1i4C&pg=PA182&lpg=PA182&dq=nltk.corpus+treebank&source=bl&ots=Y3Gmv4OIK5&sig=ACfU3U3mg-ilIUigm3D7yN1-O2J14Kr8PQ&hl=uk&sa=X&ved=2ahUKEwiAnLW3ne3oAhUEtosKHR4XC_oQ6AEwD3oECAsQLA#v=onepage&q=treebank&f=false

22. Language Tags and Ranges in the Lightweight Directory Access Protocol (LDAP) [Електронний ресурс] - режим доступу до ресурсу: https://www.ietf.org/rfc/rfc3866.txt

23. Bird Steven. *Natural Language Processing with Python* / Bird Steven, Edward Loper and Ewan Klein: O'Reilly Media Inc, 2009 - ст. 97 - 102.